

Method and means providing static dictionary structures for compressing character data and expanding compressed data.

Patent Number: ☐ EP0595064, A3

Publication date: 1994-05-04

Inventor(s): PLAMBECK KENNETH ERNEST (US); KURTZ CLARK (US); SINHA BHASKAR (US); IYER BALAKRISHNA RAGHAVENDRA (US)

Applicant(s): IBM (US)

Requested Patent: ☐ JP6222903

Application Number: EP19930115998 19931004

Priority Number (s): US19920968631 19921029

IPC Classification: G06F15/401


EC Classification: G06F17/30A, H03M7/30Z2

Equivalents: JP2502469B2, ☐ US5442350

Cited Documents: EP0375221; EP0350281; EP0582907

Abstract

Significantly improves the performance of a processor executing the well-known Ziv-Lempel (ZL) data compression/expansion algorithm by providing a novel structure for ZL dictionaries, and a novel SZL (static Ziv-Lempel) process for using an SZL dictionary(s) in a static manner to compress and/or expand records randomly accessed from a data base without spending processing on modifying the dictionary. An SZL dictionary is generated by a pass over a data base (sampling some or all of the records in the data base) before the dictionary is used for compression/expansion. (This is unlike the adaptive Ziv-Lempel, AZL, dictionary in the prior art which is generated while compressing data, during which it is continuously being "adapted" (changed) by the data it is compressing.) Entries in an SZL dictionary have novel internal structures for reducing processing time, in which a single entry may include extension characters, plural child characters or sibling characters, along with information fields and unique control fields that enable compression or expansion determinations to be made without accessing other entries involved in the determinations, which enables the SZL process to have fewer memory accesses per compression symbol for faster compression and expansion operations. Compression/expansion operations are also speeded up by accessing multiple characters per operation, enabled by matching the size of each dictionary entry to the data unit size accessed from the computer memory. The novel SZL process operates most efficiently with a separate compression dictionary and a separate expansion dictionary. However, a single SZL dictionary

may be constructed to provide both compression and expansion. 

Data supplied from the esp@cenet database - 12

BEST AVAILABLE COPY

(19)日本国特許庁(JP)

(12)公開特許公報(A)

(11)特許出願公開番号

特開平6-222903

(43)公開日 平成6年(1994)8月12日

(51)Int.Cl. ⁵	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 5/00		H 9189-5B		
12/00	5 1 1	8526-5B		
H 0 3 M 7/30		Z 8522-5J		

審査請求 有 請求項の数21 O L (全 47 頁)

(21)出願番号 特願平5-247984

(22)出願日 平成5年(1993)10月4日

(31)優先権主張番号 9 6 8 6 3 1

(32)優先日 1992年10月29日

(33)優先権主張国 米国 (U S)

(71)出願人 390009531

インターナショナル・ビジネス・マシーンズ・コーポレーション

INTERNATIONAL BUSINESS MACHINES CORPORATION

アメリカ合衆国10504、ニューヨーク州
アーモンク (番地なし)

(72)発明者 バーラークリシュナ・ラガヴェンドラ・イ
イエル

アメリカ合衆国95133、カリフォルニア州
サンノゼ、ナッシュビル・ドライブ 3049

(74)代理人 弁理士 合田 深 (外3名)

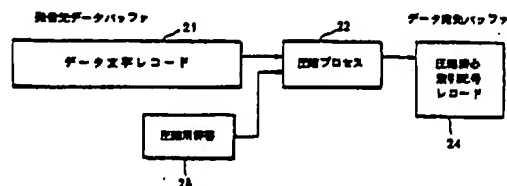
最終頁に続く

(54)【発明の名称】 文字データを圧縮し圧縮済みデータを展開するための静的辞書構造を提供する方法および手段

(57)【要約】

【目的】 Z L辞書用の新規の構造と、静的Z L辞書を静的に使用して、辞書の修正に処理を費やさずに、データベースからランダムにアクセスされるレコードの圧縮または展開等を行うための新規の静的Z L (Z i v - L e m p e l) プロセスを提供すること。

【構成】 静的Z L辞書内の項目は、処理時間を短縮するための新規の内部構造を有する。この構造では、単一の項目が、拡張文字と、複数の子文字または兄弟文字を、情報フィールドおよび固有の制御フィールドと共に含むことができる。判断手順に關与する他の項目にアクセスせずに圧縮または展開に關する判断ができるようになり、S Z Lプロセスにおいて1圧縮記号当たりのメモリ・アクセス回数を削減し、圧縮操作および展開操作の速度を上げることができる。この新規の静的Z Lプロセスは、別々の圧縮用辞書と展開用辞書を使用する場合に最も効率的に動作する。



下にある子リスト中に、アルファベット項目でも非アルファベット項目でもなく、前記親中の利用可能なスペースに、前記親中のCCと区別するために兄弟文字(S C)と呼ばれるその当該の各識別CCを含めることができない、前記同一の親の子項目(子)と関連するCCを含む、兄弟記述子(S D)と呼ばれる項目を置き、各S Cが前記同一の親の関連する子項目(子)に割り当てられた前記第1のECであるか、または前記同一の親の関連する子に割り当てられた1組の前記第1 ECであることが事前決定され、あるいは前記S D中で指示されており、それによって、前記S Cに割り当てられた前記辞書項目への追加アクセスなしで、レコード文字列の終りを検出できるようにすることにより、前記S Cが前記辞書内のレコード文字列の検出効率を向上できるようにするステップと、

前記S Dに含まれるS Cの数を示すカウント標識を前記S D中に置くステップと、

前記S D中の各S Cを、前記子リスト中の当該の子項目と関連付けられるように割り当てるステップとをさらに含む請求項7記載のレコード圧縮方法。

【請求項10】レコード文字列中の第1の文字を表す辞書項目から辞書に入り、未圧縮レコードから順次得られる後続の各レコード文字またはレコード文字の文字列を同一の辞書文字列中の後続の辞書項目と比較して前記レコード文字列の終りを決定することにより、前記未圧縮レコード中の各レコード文字列を検出するコンピュータ・プログラムを実行し、前記辞書文字列中の最後の前記項目を検出するか、あるいは前記辞書文字列中のECと一致するレコード文字の後に続く次のレコード文字と一致しない前記文字列中のECを検出することにより前記レコード文字列の終りを位置指定するステップをさらに含む請求項2記載のレコード圧縮方法。

【請求項11】比較的大規模なデータベースの対応する未圧縮レコードを再構築するために、請求項10に記載の方法によって生成される圧縮済みレコードを展開するための圧縮済みレコード展開方法であって、

Z Lアルゴリズムを使用するコンピュータ・プログラムを、前記圧縮辞書の構築に使用するデータベースに適用することによって、前記データベースの未圧縮レコードを再構築する前に、静的展開辞書を事前生成しておき、前記圧縮辞書内の辞書文字列に対応する辞書文字列を表す辞書項目を前記展開辞書内で構築するステップと、

圧縮済みレコード中にありかつ未圧縮レコード中のレコード文字列を表す各索引記号と関連する1つまたは複数の展開辞書項目を見つけることによって、前記索引記号を検出する別のコンピュータ・プログラムを実行するステップと、見つかった展開辞書項目から各索引記号で表される各レコード文字列用の文字にアクセスするステップと、

コードを再構築するステップとによって、前記圧縮辞書および前記展開辞書が事前生成された後に変更された未圧縮レコードから前記データベース内の圧縮済みレコードが生成されたかどうかにかかわらず、前記展開辞書を変更せずに、入力された圧縮済みレコードから未圧縮レコードを再生成するステップとを含む圧縮済みレコード展開方法。

【請求項12】索引記号で位置指定された前記項目内に含まれるよりも多くの先祖文字が関連する前記辞書文字列中に存在するとき、索引記号で位置指定された前記項目に連鎖された、前記展開辞書内の1つまたは複数の先祖辞書項目を提供するステップと、

前記索引記号で位置指定された前記項目中に、先祖項目によって表される1つまたは複数の先祖ECを複製して、関連する先祖項目にアクセスせずに、前記文字列中の先祖ECを出力できるようにするステップとをさらに含む請求項11記載の圧縮済みレコード展開方法。

【請求項13】Z i v - L e m p e l (Z L) アルゴリズムを使用して、比較的大規模なデータベース内で未圧縮レコードを圧縮する方法と、圧縮済みレコードを未圧縮レコードに展開する方法を組み合わせた方法であって、

データベースに前記Z Lアルゴリズムを使用するコンピュータ・プログラムを適用して、前記データベース内のすべてのZ L文字列を辞書文字列として含む辞書を前記データベースから生成することにより、前記データベースのレコードに対して圧縮操作と展開操作の両方を実行するための辞書項目を含む静的辞書を事前に生成するステップと、

別のコンピュータ・プログラムを実行して、未圧縮レコード中の一連の文字を前記辞書中の前記辞書文字列と突き合わせることににより、前記未圧縮レコード中のレコード文字列を検出するステップと、前記辞書中の辞書文字列と突き合わされるレコード文字列の、前記辞書内での終了位置を表す索引記号を出力して、前記未圧縮レコードに対応する圧縮済みレコードを提供するステップとにより、前記データベース内の未圧縮レコードが変更されているか否かにかかわらず、前記辞書を変更せずに、アクセスされた未圧縮レコードから圧縮済みレコードを生成するステップと圧縮済みレコード中にありかつ未圧縮レコード中のレコード文字列を表す各索引記号と関連する辞書項目を見つけることによって、索引記号を検出する別のコンピュータ・プログラムを実行するステップと、

見つかった辞書項目から各索引記号で表される各辞書文字列用の文字にアクセスするステップと、圧縮済みレコードから入力される各索引記号用の前記アクセスされた文字を出力して、対応する未圧縮レコードを再構築するステップとによって、前記辞書が事前生成された後に変更された未圧縮レコードから前記データベース内の

前記辞書を変更せずに、入力された圧縮済みレコードから未圧縮レコードを再生成するステップとを含む前記方法。

【請求項14】前記データベース内の各文字列を、各項目に前記辞書文字列中の1つまたは複数の当該の拡張文字(EC)が割り当てられ、かつ前記割り当てられたECが前記辞書文字列の真拡張文字(TEC)と呼ばれ、1つまたは複数の辞書項目によって表される辞書文字列として構造化するステップと、

前記辞書中の各辞書文字列中の前記辞書項目を順方向および逆方向に連鎖するステップとをさらに含む請求項13記載のプロセスを使用する方法。

【請求項15】それぞれ前記辞書文字列中に1つまたは複数の先祖ECを有する子ECを前記辞書文字列中に有する次の辞書項目を位置指定するために、辞書文字列の少なくとも第1の項目に子ポインタを記憶するステップをさらに含む請求項14に記載のプロセスを使用する方法。

【請求項16】第1の辞書項目で表されるアルファベットECの値によって示される位置に各辞書文字列の各第1辞書項目を位置指定し、各第1辞書項目をアルファベット項目として指定し、他のすべての辞書項目を非アルファベット項目として指定するが、各アルファベット項目の位置が、割り当てられたアルファベットECと関連付けられているためにどのアルファベット項目中にもECを必要としないステップと、

前記静的辞書を、ECおよび制御フィールドを含む固定長項目を含む構造にする構造化ステップとをさらに含む請求項15に記載のプロセスを使用する方法。

【請求項17】前記構造化ステップが、アルファベット項目または非アルファベット項目の前記制御フィールドを、前記親項目に割り当てられたECの後に続く、辞書文字列中の文字である子EC(CCと呼ぶ)をいつ前記親項目が含むかを示す子指標を含む親辞書項目(親項目)として構造化し、それによって、CCに割り当てられた前記子辞書項目への追加のアクセスなしで、レコード文字列の終りを検出できるようにすることにより、CCが前記辞書内のレコード文字列の検出効率を向上できるようにするステップをさらに含む請求項16に記載のプロセスを使用する方法。

【請求項18】1つまたは複数の子辞書項目(子項目)を有する子リストを含む、親項目の子辞書項目を提供するステップと、

前記子リスト中の前記第1の子項目に対して所定の位置に、前記子リスト中の各子項目を位置指定するステップと、

前記親項目中の各CCを前記子リスト中の当該の子項目に割り当てるステップとをさらに含む請求項17に記載のプロセスを使用する方法。

中の子ポインタ・フィールドによって位置指定するステップと、

現在検出中のレコード文字列用の索引記号を生成するために、子ポインタと、前記辞書文字列中のECと突き合わされる次のレコード文字と一致する前記親項目中の関連CCの前記位置とから、子辞書項目の位置を算出するステップとをさらに含む請求項18記載のプロセスを使用する方法。

【請求項20】親項目の制御フィールド中のmore-children指標をオンにセットして、前記親項目内に含まれるCCの数よりも多くの子項目が前記親項目の子リスト中にあることを示すステップをさらに含む請求項18記載のプロセスを使用する方法。

【請求項21】more-children指標を含む親項目(親)の下にある子リスト中に、アルファベット項目でも非アルファベット項目でもなく、前記親中の利用可能な前記スペースに、前記親中のCCと区別するために兄弟文字(SC)と呼ばれるその当該の各識別CCを含めることができない、前記同一の親の子項目(子)と関連する前記CCを含む、兄弟記述子(SD)と呼ばれる項目を置き、各SCが前記同一の親の関連する子項目(子)に割り当てられた前記第1のECであるか、または前記同一の親の関連する子に割り当てられた1組の第1ECであることが事前決定され、あるいはSD中で指示されており、それによって、SCに割り当てられた前記子辞書項目への追加アクセスなしで、レコード文字列の終りを検出できるようにすることにより、SCが前記辞書内のレコード文字列の検出効率を向上できるようにするステップと、

前記SDに含まれるSCの数を示すカウント指標を前記SD中に置くステップと、

前記SD中の各SCを、前記子リスト中の当該の子項目と関連付けられるように割り当てるステップとをさらに含む請求項20に記載のレコード圧縮方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本願発明は、データベースに必要な記憶スペースを大幅に削減し、データ・レコードの伝送速度を増大させ、圧縮済みデータベースに記憶された圧縮済みレコードへのランダム・アクセスを可能にし、圧縮および展開(expansion)に必要なコンピュータ資源を削減する。これらの改善は、周知のZiv-Lempel(ZL)タイプのアルゴリズムを実施するために静的辞書を固有の構造にする新規の方法によって提供される。データ・レコードを迅速に圧縮し伝送できるようにすることによって、データ・レコードの通信速度が大幅に増大する。未圧縮レコードに使用する伝送と比べてバイト伝送速度が増大しない場合でも、レコード速度が増大する。辞書の項目中に固有の指数文字構造を使用す

コードを圧縮するための処理時間と、レコードを圧縮し記憶するためのアクセス時間と、圧縮済みレコードの展開時間が短縮する。

【0002】

【従来の技術】適応Ziv-Lempel (ZL) 辞書構造または動的ZL辞書構造は、従来技術で周知である。この構造では、各入力データ・ファイルごとに適応辞書が固有に生成され、ファイルの圧縮/展開が可能になる。適応辞書は、データ・ファイルの圧縮中に構築される。展開の場合は、圧縮済みデータの展開中に構築される。辞書が一杯になると、新しい項目が、辞書内の最も以前に使用された (LRU) 既存の項目に置き換わる。

【0003】このように、従来のZLデータ圧縮方法では、辞書を、それぞれのデータ・ファイルに対して「動的に」調整することによって「適合」させていた。したがって、従来のZL辞書は、「動的」辞書または「適応」辞書と呼ばれている。従来、適応辞書がそれに関連するデータ・ファイルによって生成されない場合、データ圧縮が不十分になると考えられていた。

【0004】適応 (動的) Ziv-Lempel 圧縮辞書は、米国特許第4814746号に記載されている。この米国特許は、ZL圧縮辞書を新規データ・ファイル (すなわち、データ文字列) に適合させるための置換技術について記載している。

【0005】圧縮すべきデータが少数の孤立した記号 (アトミック記号) (バイト) から構成されているときは、適応Ziv-Lempel方式では圧縮がほとんど得られない。なぜなら、データ量が少ないので、データに関する情報が極めて少ないからである。したがって、データの圧縮を開始する前の、データに関する知識に頼るしかない。データベースを構成する個々のレコードを他のレコードとは独立に圧縮または展開する必要がある、データベースを圧縮する場合がこれに当てはまる。このように圧縮または展開する必要があるのは、ツリー構造インデックス方式またはハッシュ・アクセス方式を介して単一のレコードまたは少数のレコードに直接アクセスしなければならないからである。レコードが容易に見つかり、見つかったら、そのレコードに関するデータだけが展開されることが望ましい。レコードは通常小さく、小さいレコードに対して適応Ziv-Lempel圧縮を実行しても、圧縮が完了する前にごく少数の短いZL辞書文字列しか形成できないため、それほど圧縮は行われない。

【0006】

【発明が解決しようとする課題】本願発明は、データを圧縮し展開するための静的辞書構造を定義するものである。プログラムが、データベース上でバスを1回行い、データベース内の一部または全部のデータをサンプリン

このデータベースを使用すると、本願発明の別の一部を構成する方法によってレコードを圧縮または展開することができる。

【0007】本願発明の新規の静的ZL辞書は、データ・レコードの効率的な圧縮処理および展開処理を可能にする固有の構造になっている。本願発明は、大型データベース中のレコードから生成された静的ZL辞書を使用する。本願発明では、辞書を、現在伝送または記憶するために圧縮中のレコードに「適合」させない。静的辞書は、データベース変更後でも、圧縮中の未圧縮レコードの変更を含めて、変更なしで引き続き使用することができる。

【0008】静的辞書は、送信側位置と受信側位置のどちらに配置することもでき、データ・レコードと共に伝送する必要はない。送信元では、辞書が圧縮に使用される。圧縮されたデータは宛先に送信され、宛先では、同一のディレクトリのコピーを使用して、圧縮済みデータが展開される。レコード伝送速度は、1レコード当たりのビット数が減るにつれて増大し、1レコード当たりのビット数が増えるにつれて減少する。したがって、データ圧縮によって1レコード当たりのビット数が削減できるなら、伝送回線の伝送速度が変更されていなくても、データ・レコードの伝送速度を上げることができる。

【0009】本願発明では、圧縮済みデータだけを伝送すればよいので、データ・レコードの伝送速度が大幅に増大する。

【0010】本願発明は、データ・レコードの圧縮プロセスまたは展開プロセスの速度を上げる、ZL辞書中の項目用の新規の内部構造を提供する。この新規項目構造によって、単一の項目に、複数の文字と、情報フィールドおよび固有の制御フィールドとを含めることができるようになる。このため、圧縮についての決定に関与する項目にアクセスする前にその決定を下せるようになり、その結果、1圧縮記号当たりのメモリ・アクセス回数が削減され、圧縮操作の速度が上がる。また、1操作当たり複数の文字を処理することによって展開操作の速度も上がる。

【0011】本願発明はまた、別々の圧縮用辞書と展開用辞書を使用することによって、圧縮プロセスおよび展開プロセスの速度を増大させる。このため、辞書項目の内部のスペースを効率的に使用することができる。圧縮と展開の両方に単一の辞書を使用する場合、このような効率的な使用は不可能である。

【0012】記憶スペースが重要である場合、本願発明で単一の静的辞書を圧縮と展開の両方に使用して、処理の速度を上げることができる。ただし、別々の辞書を使用すれば、最高の操作速度が得られる。

【0013】したがって、静的圧縮用辞書は、未圧縮レコードを記憶または送信する必要がある送信側位置に格

回線、マイクロ波、ファイバ回線など任意の種類のデジタル伝送手段によって遠方のコンピュータ・システムから圧縮済みデータ・レコードを受信して処理する位置に提供される。ある位置で送信と受信のどちらか一方だけを行なう場合、その位置には静的圧縮用辞書か静的展開用辞書のどちらか一方だけが必要であり、両方とも必要なのではない。したがって、本願発明では、圧縮および展開を別個の独立したプロセスとして扱うことができる。

【0014】さらに、本願発明は、コンピュータ・システムのハードウェア特性に適合した構造の静的ZL辞書を提供することによって、コンピュータ・システムによるデータ・レコードの圧縮速度または展開速度を大幅に向上させる。これは、辞書中の項目のサイズを、コンピュータ・メモリからアクセスされるデータ・ユニットのサイズと一致させることによって行う。

【0015】本願発明の静的辞書は、データベース中の様々なレコードの圧縮または展開に使用している間には変更されない。しかし、静的辞書がまったく変更できないわけではない。静的辞書のデータベースは静的である必要はなく、通常、変更可能である。経験によれば、大型データベースでは新しいレコードがたえず追加および削除されるので、一般に、データベースは時間の関数として少しだけ変化する。本願発明は、データベースが少しだけ変化した場合でも、圧縮や展開の効率にほとんど影響なく、静的辞書を継続して使用できるという、事実を利用している。

【0016】この事実は、大型データベースは比較的長い期間かけて比較的ゆっくりと変化するもので、データのZL文字列がほとんど変わらず、未変更の静的圧縮用辞書を使って引き続き高い圧縮効率を得ることができるという認識に基づいている。データベースにたとえば10%を上回るような大幅な変更が生じる場合、その時点で存在しているデータベース用の静的辞書を再生成することができる。本願発明は、文字列を終了する判断文字を辞書中の前の項目に移して、圧縮プロセスで圧縮記号を判定するためのメモリ・アクセス回数が減るように項目を構造化し、圧縮操作の速度を上げる。

【0017】上述の利点を実現可能にする本願発明の諸特徴を要約する前に、ZL方式に関する情報と、本願発明がZL方式とどう関係しているのかについてさらに説明する。

【0018】既知の適応辞書の場合と同様に、本願発明の単一の辞書または別個の辞書はツリー形式の構造になっている。このツリーは、下向きに成長するツリーとして視覚化することができる。ツリーの1番上の256のノードまたは項目は、アルファベット項目と呼ばれ、8ビット・バイトの256の可能な値に対応する。各アルファベット項目は、子項目を持つ親項目となる。それが

子項目を持つ親項目となることができ、以下同様である。

【0019】適応辞書または本願発明の辞書では、各項目は、1または複数の拡張文字(expansion character)を表す。この拡張文字は、その項目のすべての先祖によって表されかつZL文字列を形成するあらゆる拡張文字の右側に続く。ZL文字列を文字記号と呼ぶ。したがって、各辞書項目は1つまたは複数の拡張文字のみならず、今定義した完全な文字記号をも表す。

10 【0020】適応辞書または本願発明を使用し、入力文字列の文字を、辞書中の拡張文字と、一致する文字が見つかるまで左から右へと比較していくと圧縮が行われる。次に、最後に一致する辞書項目の拡張文字によって最後の可能な一致が見つかる、索引記号と呼ばれる、その項目の索引が、圧縮済みデータとして出力される。圧縮プロセスは次の文字に進む。この文字は、レコードから入力された文字シーケンスの最初の一致しない文字である。

20 【0021】上述より、索引記号は、文字記号(ZL文字列)を表す辞書項目を識別し、したがってその文字記号を識別することが分かる。

【0022】現代の大部分のコンピュータ・システムは2進法を使用しているので、これらのシステムでは、辞書中の項目の数は、かならず2の累乗値となる。特定の辞書の索引記号中のビット数が、そのべき指数であり、この数は、索引記号によって識別される文字記号中のビット数よりはるかに小さいことが多く、あるいはそれが通常である。このため、所望の圧縮を実現することができる。

30 【0023】適応辞書または本願発明を使用し、一連の索引記号を左から右へと処理していき、各索引記号を使用して辞書項目を見つけて、その項目で表される文字記号を出力することによって、展開が行われる。適応辞書を使用するときは、見つかった辞書項目と、その項目の各先祖から拡張文字を取り出すことによって、文字記号が再構築される。本願発明は、従来よりも少数の辞書項目にアクセスすることによって再構築が行える新規の手段を含む。

40 【0024】本願発明の説明の最初に、圧縮と展開の両方に用いる単一の静的辞書に適用される本願発明の多数の特徴を紹介し明らかにする。次に、好ましい実施例、すなわち別々の静的圧縮用辞書および静的展開用辞書を示すことによって、本願発明をさらに詳しく説明する。本発明者等は、単一辞書に適用した場合の上記特徴の研究を通じて、好ましい実施例を発明した。

【0025】単一の辞書を使用する際に本願発明で提供される特徴には、次のものがある。

【0026】1. 辞書項目を固定した2の累乗の長さにした。項目を連続した記憶域に置く。これによって、索引

迅速に見つけることができる。このように辞書項目を見つけるには、索引記号の右側に一定数のゼロ・ビットを付加し、次にその結果を辞書の開始アドレスに加算して、識別された項目のアドレスを形成する。コンピュータ・システムのデータ転送機能と、項目中で所望される情報の種類および量に基づき、8バイト長が最適の長さであることが分かっている。

【0027】2. アルファベット項目中では、その項目で表される拡張文字 (EC) を省略する。これによって、項目中のスペースが節約され、このスペースを項目中で他の情報に使用することができる。

【0028】3. アルファベット項目以外の項目中に、項目のすべての先祖に含まれる EC の右側に付加されるとみなされる1つまたは複数の EC を置く。項目中に複数の EC を置くと、複数の項目ではなく単一の現項目だけにアクセスすることによって、入力文字列中の複数の文字を比較することができ、また辞書中の情報密度が高くなり、長い ZL 文字列を所定のある数の項目で表すことができるので、圧縮が向上する。複数の EC のうち最初の EC より後の EC を追加 EC (AEC) と呼ぶことがある。

【0029】4. アルファベット項目以外の項目中に、その項目によって ZL 文字列中で新たに表される EC のみならず、可能なら、その項目の最も近いある数の先祖によって表されるある数の EC も置く。先祖によって表される EC を先祖 EC (PEC) と呼ぶことがある。ある項目によって新たに表される EC を真 EC (TEC) と呼ぶことがある。ある項目中の PEC は、その項目の TEC の左側に配置される。項目中に PEC を配置すると、展開時に、先祖項目にアクセスせずに現項目から PEC を取り出すことができる。

【0030】5. アルファベット項目以外の項目中に、可能なら、その項目の先祖のポインタを置くことができる。このポインタは、先祖ポインタ (PPTR) と呼ばれ、先祖の索引であり、展開時に使用される。先祖は、現項目の親でも、さらに遠い先祖でもよい。現項目によって表される文字記号が非常に大きくて、現項目中に記号内のすべての EC を含めることができないとき、PPTR が必要である。さらに、PPTR で指定される項目も PPTR を含むことができる。展開時に文字記号を再構築する際、PPTR を使って一連の辞書項目を上向きに進まなければならない場合がある。本願発明では PEC および PPTR を新たに使用するので、通常、索引記号の展開時には、圧縮時に索引記号を形成する際にアクセスしなければならなかったよりも少数の辞書項目にアクセスするだけで済む。

【0031】6. アルファベット項目以外の項目中に、その項目と、その項目のすべての先祖における EC の総数のカウントを含む記号長 (SL) フィールドを置く。

号の長さをただちに知ることができ、出力バッファ中に記号用のスペースがあるかどうかを判定することが可能になる。EC が、現項目の PPTR によって指定される項目中に EC として現れる場合、それは現項目中に PEC として配置されないという規則を採用すると、SL の使用に関して次の規則が利用可能になる。項目が PPTR を含まない場合、その項目の SL はその項目中の EC の数を指定する。項目が PPTR を含む場合は、その項目の SL から、PPTR で指定される項目の SL を引いた数が、その項目中の EC の数を指定し、その項目の SL からその項目の親の SL を引いた数が、その項目中の TEC の数を指定する。親がアルファベット項目である場合、その暗黙的に SL が 1 に等しい。単一圧縮および展開辞書における項目を構成する本願発明の方法によって、項目は、その SL が 7 以下で子を持たない場合、そのあるいは SL が 5 以下で子を持つ場合、PPTR を含まないことが分かる。

【0032】7. 記憶域内の連続する項目のリストに親の子供を置き、その親中の子ポインタ (CPTR) によってリストの第 1 項目を指定する。CPTR は、第 1 の子の索引である。子供が連続しているので、次の子を見つけるとき、その子中に制御情報が必要とされないの、子中のスペースが節約され他の情報用に使用できるようになる。

【0033】8. 値が 1 のとき、項目が子を持ち、したがって CPTR を含むことを示すビットを項目中に置く。このビットを子ビット (C) と呼ぶ。このビットによって、CPTR が必要とするスペースを、CPTR が存在しないとき他の情報に使用することができる。また、このビットによって、それがゼロのとき、現項目に子がないので突合せプロセスが終了することをただちに知ることができる。

【0034】9. ある数のある親の初期の子のそれぞれの第 1 TEC の複製を親中に置く。この場合、親中のこれらの複製 EC のそれぞれを子文字 (CC) と呼ぶ。親中の CC によって子にアクセスして、子中にある同一の EC を取り出す必要はなく、これらの CC を入力文字列の次の文字と比較することができる。対応する子項目にアクセスして突合せプロセスを継続する必要があるのは、CC が等しい場合だけである。CC が一致する場合、対応する子の索引は、項目中の CPTR と、項目中の 1 組の CC のうち一致する CC の位置番号とから計算することができる。項目中で CC を含めるために利用可能なスペースの量は、その項目中の EC (PEC および TEC) の数と、その項目が CPTR または PPTR を含むかどうかによって決まる。CC が一致しても、対応する子に一致が見られるかどうかはまだ分からない。なぜなら、その子は、CC と同一な TEC を 1 つだけ含むのではなく、2 つ以上の TEC (1 つ以上の AEC) を

【0035】10. 項目中に、値が1のとき、その項目が、そのCCが表す子よりも多数の子を持つことを示すビットを置く。このビットは、more-childrenビット

(M)と呼ばれ、項目のCCで表される子に、CCおよびAECによって一致が見られないときに使用される。そのような一致がなく、Mが1であるときは、その項目のCCで表される最後の子の次の最初の子にアクセスし、その最初の子に第1TECおよびAECがあれば、それを入力文字列中の次の文字と比較する必要がある。

【0036】11. 項目中に、値が1のとき、その項目が、現行の子リスト中でその項目の後に続く兄弟を持つことを示すビットを置く。ただし、2つの項目がどちらも同一の親の子である場合、それらを兄弟と呼ぶ。このビットは、more-siblingsビット(S)と呼ばれ、その項目の親のMビット、または先行の兄弟もしくは兄弟記述子中のSビットのためにアクセスできる項目中だけで使用される。兄弟記述子については後述する。

【0037】12. 現項目の後に続くある数の兄弟のそれぞれの第1TECの複製を現項目中に置く。この場合、現項目中のこれらの複製ECのそれぞれを兄弟文字(SC)と呼ぶ。ある項目中にSCがあると、兄弟にアクセスして、その中にある同一のECを取り出す必要はなく、これらのSCを入力文字列の次の文字と比較することができる。対応する兄弟にアクセスして突合せプロセスを継続することが必要になるのは、SCが等しい場合だけである。SCが一致する場合、対応する兄弟の索引は、現項目の索引と、その項目中の1組のSCのうち、一致するSCの位置番号とから計算することができる。SCが項目中に置かれるのは、その項目に子がない場合だけである。SCは、項目に子がある場合にCCを格納するのに使用されるスペースに配置される。

【0038】13. CCまたはSCの数が項目中に配置できる最大数よりも少ないとき、項目中の最後のCCまたはSCに等しい文字を最後のCCまたはSCの右側に配置することによって、項目中のCCリストまたはSCリストの終りを示す。これによって、項目中のスペースを、項目中のCCまたはSCの数の明示的なカウントを含めるため専用にする必要がなくなるので、スペースを節約して他の情報用に使用することができる。この特徴では、ある親の子供のどの第1TECも同一でないことが必要である。

【0039】14. 後続の兄弟に対応するSCだけを子リストに記憶するために、子リスト中に兄弟記述子(SD)と呼ばれる特殊な種類の項目を配置する。SDではない辞書項目は現在、SDと区別する必要があるときは文字項目(CE)と呼ばれる。SDが、普通ならCEを記憶するのに利用できる辞書索引位置を占めることに留意されたい。SDの目的は、親にCCを格納するスペースがない場合、または先行の兄弟にSCが格納されてい

Cまたは先行の兄弟中のSCと同様に、SD中にSCがあると、後続の兄弟にアクセスする必要がなく、入力文字列中の次の文字を、後続の兄弟の第1TECと比較することができる。したがってアクセスが不要になると共に、圧縮プロセスの速度が上がる。SDは、2種類の項目のそれぞれにおける制御フィールドの特殊な値によってCEと区別される。SDの存在は、親のMビット、または先行の兄弟中もしくはSD中のSビットによって示される。SDは、CEと同様に、Sビットを含むことができる。子リストは、まずある数の子、次にSD、次にSD中のSCで示される兄弟、次に別のSDなどという順序に並べることができる。SD中のSCリストの終りは、CEの場合と同様に、あるいはSLフィールドの値によって示することができる。SLはSD中では通常の方法で使用されないからである。

【0040】単一辞書に関する、本願発明の特徴に関する説明をこれで終わる。この説明で使用し、今後も頻繁に使用する略語を容易に覚えられるように、以下に列挙しておくことにする。

【0041】AEC 追加拡張文字。項目中の第1TECの次のTEC。

【0042】Cビット 子ビット。項目が子を持ち、CPTRを含むことを示す。

【0043】CC 子文字。親中の文字であり、対応する子中の第1TECと位置が同一である。

【0044】CE 文字項目。SD以外の辞書項目。単に「項目」という語を使用するときは通常、文字項目を意味する。

【0045】CPTR 子ポインタ。項目の最初の子の索引。

【0046】EC 拡張文字。項目中の文字であり、その項目で表されるZL文字列の一部である。

【0047】Mビット more-childrenビット。項目中のCCの数よりも多数の子が項目にあることを示す。

【0048】PEC 先祖EC。現項目中の文字であり、現項目の親またはそれよりも遠い先祖中のECでもある。現項目中のPECは、その項目中のTECよりも前に置く。

【0049】PPTR 先祖ポインタ。項目の先祖の索引。

【0050】Sビット more-siblingsビット。項目中のSCの数よりも多数の兄弟を項目が持つことを示す。項目はCEまたはSDとなることができる。Sが1である場合は、後続のCEまたは後続のSDが存在することを示す。

【0051】SC 兄弟文字。項目中の文字であり、対応する兄弟中の第1TECと位置が同一の文字である。

【0052】SD 兄弟記述子。CE以外の辞書項目。SDは、制御フィールドの特殊な値によって区別される。

の先祖中のECの総数のカウント。

【0054】TEC 真EC 先祖項目中には現れないEC。すなわち、PECではないEC。

【0055】本発明者等は、次のような、単一の辞書を圧縮と展開の両方に使用する方法の欠点を発見した。

【0056】項目中にPECを置くと、展開性能は向上するが、CCまたはSC用のスペースが少なくなり、あるいはまったくなくなるので、圧縮性能は低下する。したがって、PECが存在すると、辞書は展開性能に有利となり、PECが存在しないと、辞書は圧縮性能に有利となる。圧縮中にも展開中にも最高の性能が得られることが望ましい。

【0057】現項目中のSLと現項目のPPTRで指定される項目中のSLと現項目の親のSLとに基づいて現項目中のECおよびTECの数を求めるための計算は時間がかかり、圧縮および展開の性能が低下する。

【0058】TECの複製を、CCとして親中に、またはSCとして兄弟もしくはSD中に置くと、辞書中に同一の文字が2回存在することになり、他の情報用に利用できるスペースが減少する。

【0059】最後のCCまたはSCに等しい文字によって、CE中のCCリストもしくはSCリスト、またはSD中のSCリストの終りを示す技法では、文字の比較を繰り返す必要があり、CCまたはSCの数の単純なカウントを使用するよりも時間がかかる。

【0060】ある親の子供のどの第1TECも同一であってはならないという、親中の最後のCCまたは兄弟もしくはSD中の最後のSCを示す手段を提供するための要件により、様々な数の同一文字が繰り返される長い文字列の場合、有効な辞書構造が得られない。このような文字列では、2つ以上の子が同一のTECで始まること

が望ましい。

【0061】本発明者等は、項目中の制御情報が次のように配列されている場合、辞書項目中でPEC、TEC、CC、およびSC用に最大のスペースを利用できることを発見した。Cビット、Mビット、およびSビット、さらに5ビットのSLを1バイト内に配置する。ただし、アルファベット項目はSもSLも含まないので、CおよびMはアルファベット項目中の4つのビット内に配置することができる。また、CPTRとPPTRはそれぞれ別々の12ビット・フィールド内にあるが、このフィールドは項目中で必要なことも必要でないこともある。5ビットのSL長により、最大文字記号が32文字に制限される（SLで表される値がSLフィールド中の物理値より1だけ多くなる、1増し表記法が使用される）。12ビットのCPTR長またはPPTR長により、辞書中の項目の最大数が4K（4096）に制限される。32および4Kというこれらの限界はどちらも過度に小さいものではないが、それでも問題となる可能性

【0062】上記の欠点は、以下に示す本願発明の別の特徴によって解消される。これらの特徴には、列挙された欠点に特に対処するためのものではない新しい特徴も含まれる。以下に示す特徴には、上記の特徴の番号に続く通し番号が付けてある。

【0063】15. 単一の辞書の代わりに別々の圧縮用辞書と展開用辞書を提供する。これによって、各辞書の内容をそれぞれの目的に合わせて最適化し、それによって圧縮時にも展開時にも最高の性能を実現することができる。圧縮用辞書は、単一辞書の場合よりも多くのPECを含むことができる。圧縮用辞書はまた、単一辞書の場合よりも多くのCCを含むことができる。SDは依然として、SCを含めるため、および圧縮用辞書CEの取扱いを単純にするために利用可能なので、CEには含まれず、SDだけに含まれる。したがって、CE中のmore-siblings (s) ビットは削除される。この場合も、圧縮用辞書および展開用辞書のそれぞれにおける最適長は8バイトであることが分かっている。したがって、それぞれの辞書に同数の項目を入れるものとする、別々の辞書を使用する場合、単一の辞書に必要なスペースの2倍のスペースが必要になる。性能よりもスペースを優先したい場合は、別々の辞書ではなく単一の辞書を使用することができる。

【0064】16. 展開辞書項目中のSLフィールドを除去し、部分記号長(PSL)フィールドおよび完全記号長(CSL)フィールドで置き換える。項目中でPSLがゼロのときは、その項目で表される文字記号は、完全にその項目内に含まれ、項目のCSLで指定される長さを持つ。文字記号はPECとTECから構成することができ、TECの前にPECがいくつあるかを決定する必要はない。展開時には、文字記号を項目から取り出し、出力バッファの次のバイト位置から配置していただくだけでよい。PSLがゼロである項目を、先行なし項目と呼ぶ。項目中でPSLが非ゼロのときは、項目内に、その項目で表される文字記号の一部だけが含まれる。項目中には、PECまたはTECである、文字記号の1番右の文字がPSLの数だけあり、1番右の文字の左側にある残りの文字は、現項目のPPTRで最初の項目が識別される、1つまたは複数の項目中にある。CSLフィールドは現項目中に存在せず、CSLが占めるはずのスペースは、PPTRの一部を格納するために使用される。項目は、1バイトのオフセット(OFS)フィールドを含み、その内容は、出力バッファの次のバイト位置と、現項目中のPECおよびTECの配置が開始する位置の間のバイト数である。したがって、SLを使用する時間のかかる計算が不要になる。また、PSLの最大値が5であり、OFSの最大値が255なので、文字記号の最大長は、32バイトではなく260バイトとなる。PSLが非ゼロである項目を、先行あり項目と呼

【0065】17. 非アルファベット圧縮辞書項目中の第1TECを除去し、このECを親中のCCまたはSD中のSCとしてだけ出現させる。これによって、項目中のスペースが節約され、他の情報用に使用できるようになる。アルファベット項目に現れなかったのは、第1TECだけである。

【0066】18. 圧縮辞書項目中のSLフィールドを除去し、項目中のAECの数のカウントを含むAECカウント(ACT)フィールドで置き換える。これによって、入力文字列中の次の文字を項目のAECと比較する際に計算が不要になる。

【0067】19. 圧縮辞書項目中の子ビット(C)を除去し、項目中のCCの数のカウントを含む子カウント(CCT)フィールドで置き換える。CCおよびCPT Rが存在するのは、CCTが非ゼロの場合だけである。CCTにより、文字比較を行ってCCリストの終りを検出する必要がなくなり、ある親の任意の数の子供が、同じ親の別の子の第1TECと同じ第1TECを持つことができるようになるので、同一の文字が繰り返される長い文字列の取扱いが改善できる。

【0068】20. 圧縮辞書項目中でCCTとACTの3種の組合せだけを使用可能にする。CCTが0の場合、ACTは0ないし4となることができる。CCTが1の場合、ACTは0ないし4となることができる。CCTが1より大きい場合、ACTは0または1となることができる。本発明者等は、この組合せの制限によって、実施態様が単純になり、実行速度が上がるが、依然として優れた辞書の圧縮機能が得られることを発見した。ACTを0または1だけになるように制限すると、ACTは単一ビットになり、項目中のスペースが節約される。この単一ビットを倍文字項目・ビット(D)と呼ぶ。

【0069】21. SD中のSDの特殊指示をなくし、SD中のSLをなくし、SD中のSCの数のカウントを含む兄弟カウント(SCT)フィールドで置き換える。SCTによって、文字比較を行ってSCリストの終りを検出する必要がなくなり、子供が同一の第1TECをもつことが完全に可能になる。

【0070】22. CEおよびSD中に、子検査ビット(XおよびY)という1組のビット置く。CEの各CC用にCE中に子検査ビット(X)があり、SDの各SC用にSD中に子検査ビット(Y)がある。子検査ビットが1のときは、そのビットに対応するCCまたはSCに一致が見られる場合、そのCCまたはSCに対応する子を検査して、突合せプロセスを継続する必要があることを示す。子がAECを持つかどうか検査する必要がある。なぜなら、AECは入力文字列の次の文字と一致することもしないこともあり、したがって子に一致が見られ、子と見られない子もあるからである。子が子供

一致が見られる場合、一致している子の子に一致が見られるかどうか調べることによって突合せプロセスを継続する必要があるからである。CCまたはSCに一致が見られ、対応する子検査ビットがゼロの場合、その子にはAECも子供もなく、したがって最適な一致が見つかったことがただちに分かる。この場合、子を検査する必要はなく、辞書へのアクセスが節約される。

【0071】23. 同一の文字が繰り返される多数の異なる長さを持つ文字列を圧縮するのに必要な記憶域参照の回数を67%減らす、圧縮辞書項目を配列する新規の方法を示す。この新規の方法は、文字用のアルファベット項目が、文字の第2のインスタンスを表す1つの子を持ち、この子が、文字の第3のインスタンスを表す1つの子を持ち、さらに、この子が文字の第4のインスタンスを表す1つの子を持つという自明な方法とは対照的である。この新規の方法は、任意の項目にAECが存在でき、親項目が、それぞれ同じ第1TECを持つ複数の子供を持つことができる(すなわち、親が複数の同一CCを含むことができる)という、本願発明の能力を活用している。この新規の方法では、アルファベット項目が第1インスタンスである奇数レベルの親は、AECを持たず、5つの子を持ち、子は順に4つ、3つ、2つ、1つ、および0のAECを持つ。子は、その親よりも5つないし1つ多くの文字を表す。奇数レベルの親の第1の子である偶数レベルの親は、4つのAECと1つの子を持ち、この子が次の奇数レベルの親となる。奇数レベルの親の2番目ないし5番目の子は、子を持たない。しかし、この方法では、追加の子が、同一の繰返し文字以外も含むZL文字列を形成するのに役立つ場合、奇数レベルの親が5つより多数の子を持つこともでき、奇数レベルの親の2番目ないし5番目の子が子供を持つこともできる。

【0072】24. 短記号文字列という特殊な形式の圧縮済みデータを提供する。この形式は、短記号オプション(SSO)というオプションを呼び出す際に提供される。SSOを呼び出すと、索引記号の前にゼロ・ビットが付き、長記号と呼ばれるようになる。圧縮済みデータは、短記号と呼ばれる未圧縮文字を含むことができる。4ビットの短記号ヘッダの後に続く文字列に、1つないし8つの短記号が入ることができる。短記号ヘッダの第1ビットは、このヘッダを長記号と区別するために1になる。残りの3ビットは、文字列中の短記号の数よりも1多いカウントを含む。圧縮時には、必ず1文字だけから成る文字記号を識別する長記号ではなく短記号が作成される。1つないし8つの短記号が連続的に生成されるときは、短記号文字列としてまとめて配置される。本発明者等は、圧縮すべきデータが、いくつかのまたは多数の2文字以上のデータ・シーケンスが辞書中にZL文字列として存在しないようなデータであるとき、この新規

ト内の2進データ、およびバイト内のバック10進データでそうなると思われる。辞書中の項目の数が少ないほど、SSOの利点は減少する。なぜなら、そのような場合、索引記号の長さに1ビット追加して長記号を形成すると、スペースがかなり使用されるからである。

【0073】別個の辞書に関する、本願発明の追加の特徴に関する説明をこれで終わる。上記の説明で使した略語を容易に覚えられるように、以下に列挙しておくことにする。

【0074】ACT 追加拡張文字カウント。圧縮辞書項目中のAECの数。

【0075】CCT 子文字カウント。圧縮辞書項目中のCCの数。

【0076】CSL 完全記号長。先行なし展開辞書項目中のPECとTECの数。

【0077】Dビット 倍文字項目・ビット。0または1のACTと同じ。

【0078】OFST 出力バッファの次の位置から、この展開辞書項目のPECおよびTECの配置が開始する位置までの、バイト単位で表したオフセット。先行あり項目中に存在する。

【0079】PSL 部分記号長。0の場合、先行なし展開辞書項目を示す。非ゼロの場合、先行あり項目を示し、項目中のPECとTECの数のカウントになる。

【0080】SCT 兄弟文字カウント。SD中のSCの数。

【0081】SSO 短記号オプション。

【0082】Xビット CC用の子検査ビット。

【0083】Yビット SC用の子検査ビット。

【0084】

【実施例】

圧縮プロセスの概要一図1

図1において、ボックス21は、未圧縮レコードを受け取るデータ発信元バッファを表す。発信元バッファ21は、それに記憶されているレコード中の文字を、各レコードに文字が記憶された順序で圧縮プロセス22に送る。コンピュータ・システムのメモリ内に置くことができる。圧縮プロセス22は、現未圧縮レコードから受け取った文字を、圧縮用辞書23中のアクセスされた文字と突き合わせて、現レコードの文字シーケンス内の文字列を検出する。圧縮プロセス22は、文字列が検出されるたびに、データ宛先バッファ24に「索引記号」を出力し、宛先バッファ24内で各圧縮済みレコードが索引記号のシーケンスとして生成される。入力文字が、辞書項目の範囲内で、現在比較されているどの文字とも突き合わせることができない場合、そのたびに、圧縮プロセス22によってある文字列が検出される。出力される索引記号は、最後に検出されたZL文字列中の最後に一致した文字の、辞書内の位置を表す。文字が一致しない場

のZL文字列について、圧縮用辞書に再度入って、この次ZL文字列中の1つまたは複数の文字を検出する。

【0085】未圧縮レコードの最後の文字が圧縮プロセス22に提供されると、このレコードの最後の文字列が終了し、この最後に一致した文字の辞書位置が、宛先バッファ24に出力され、対応する圧縮済みレコードの文字列を表す索引記号になる。

【0086】圧縮済みレコードは次に、宛先バッファ24から次の宛先に送信することができる。次の宛先はたとえば、ディスク記憶域や、通信回線を経て伝送するためのモデムなどとしてすることができる。

【0087】展開プロセスの概要一図2

図2は、圧縮済みレコードを受け取り発信元バッファ26に入れる、受信位置における圧縮済みレコードの展開処理を表している。発信元バッファ26は、コンピュータ・システムの主記憶域内の指定された区域とすることができる。圧縮済みレコードはたとえば、ディスク記憶域や、通信回線に接続されたモデムなどから受け取ることができる。

【0088】発信元バッファ26内の圧縮済みレコードを含む索引記号が、展開プロセス27に送信される。展開プロセス27では、展開用辞書28を使用するが、これは図1の圧縮用辞書23と異なっており、索引記号は、発信元バッファ26で読み取り中の現圧縮済みレコードの始めから既存の順序で展開プロセス27に送信される。

【0089】展開プロセス27は、発信元バッファ26から受け取った各索引記号を検出し、その値を展開用辞書28の項目の索引として使用して、展開辞書28内の対応する未圧縮文字列のすべての文字または1番右の文字を取り出す。選択された文字列に先行の文字があれば、展開用辞書28内の1つまたは複数の連鎖された先行の項目から取り出されて、現索引記号で表される未圧縮文字列が再構築される。選択された文字列中の文字は、宛先バッファ29に現カーソル位置から順に書き込まれていく。

【0090】圧縮用辞書および展開用辞書の構造

本願発明の静的Ziv-Lempel方式は、2つの辞書を使用することによって最高の性能を得る。これらの辞書は、コンピュータ・システムのメモリに配置することができる。本明細書では、これらの辞書の一方を「静的圧縮用辞書」と呼び、他方を「静的展開用辞書」と呼ぶ。各辞書にはそれぞれ、固有の項目構造が提供される。

【0091】未圧縮レコード中のバイトが、レコードを圧縮する際に、辞書に記録された文字列と突き合わされる。辞書内の文字列と一致する未圧縮レコード中の各文字列は、その文字列中の最後の文字の、辞書内の索引を、「索引記号」と呼ばれる圧縮コードとして使用する

字列を「文字記号」と呼ぶ。

【0092】文字列中の文字の数は、1から辞書に含まれる最大文字列長までの任意の数とすることができるので、文字記号は可変長である。

【0093】索引記号は固定長であり、この長さは、圧縮用辞書内の項目の数によって決定される。したがって、辞書が512バイト、1024バイト、2048バイト、または4096バイトの長さを持つ場合、索引記号の長さは9ビット、10ビット、11ビット、または12ビットである。索引記号は、出力される「圧縮済みレコード」中で互いに連続して配置される。

【0094】本願発明で使用する辞書は、Ziv-Lempel (ZL) アルゴリズムを中心として編成されているが、ZLアルゴリズムは多数の異なる方法で実施することができる点に留意されたい。本願発明では、そのうちで、コンピュータの性能の点で有利な、Ziv-Lempelタイプの辞書内の項目の固有の構造化を伴う新規の方法を提供する。

【0095】辞書内のどの文字列についても、その接頭文字列も辞書内にあるという点で、Ziv-Lempel アルゴリズムは接頭特性を有する。すなわち、文字列SBが辞書内にあり、ある文字列Sおよび別の単一文字Bから構成される場合、文字列Sは辞書内にあるはずである。Bを、接頭文字列Sの終りにある「拡張文字」*

未圧縮シーケンス→ABCD

圧縮済みシーケンス→374

【0100】ZL方式では、圧縮アルゴリズムを使用し、辞書内の文字列と一致する最長の未圧縮文字列を見つける。各文字列が検出されると、その文字列中の最後の文字の索引（辞書アドレス）によってその文字列が識別される。

【0101】したがって、ZL圧縮方式では、次のステップが実行される。

【0102】1. 現入力位置から始めて、辞書のメンバーである最長文字列を検索する。

【0103】2. 文字列の最後の文字の、辞書内の位置（索引）を使用することによって、その文字列を表す索引記号を出力する。

【0104】3. ステップ1に戻って、最後に検出された文字列に続く文字を次の文字列の最初の文字として文字列検出プロセスを続行する。ただし、圧縮中の文字シーケンスの終りにある場合は除く。

【0105】ZL展開アルゴリズムでは、逆のプロセスに従い、圧縮操作によって得られた索引記号のシーケンスを入力として使用して、未圧縮文字シーケンスを再構築する。説明を簡単にするため、文字列全体が辞書の各項目に記憶されるように示してあるが、好ましい実施例ではこれを避けることに留意されたい。

【0106】図4に、A、B、C、AA、AB、AB

* (EC) と呼び、文字列S中の各文字もまたECである。文字列が一時に1文字だけ拡張されるので、これを「文字拡張」と呼ぶ。ZLアルゴリズムはまた、従来技術では、接頭文字列が、記号と呼ばれる2文字以上ずつ一時に拡張される、「記号拡張」によっても実施されている。本願発明では、文字拡張方式と記号拡張方式の両方が使用可能である。

【0096】ZL辞書の最初の256項目はそれぞれ、8ビット・バイトの各ビットのあらゆる置換によって得られるすべての文字からなるアルファベットのすべての文字を含む。あらゆるZL文字列は、1つの文字だけを持つ単一文字列を含む、辞書内のこれらの最初の256文字の1つで始まる。最初の256の文字項目に続く項目中のECは、複数の文字を持つ文字列中に存在する。

【0097】Ziv-Lempel方式は、未圧縮文字のシーケンスを、固定長「索引記号」の圧縮済みシーケンスに変換する。各記号は、発信元文字シーケンス中の文字の文字列を表す。ZL方式では、入力として索引記号シーケンスが与えられると、その各索引記号をそれが表す文字列に展開し、それによって元の未圧縮文字シーケンスを再作成する。

【0098】図3に、次の例で最初の索引記号を生成するために検索される辞書項目を示す。

【0099】

未圧縮シーケンス→	ABCD	EFG	H	IJKLMN	EFG
圧縮済みシーケンス→	374	442	200	996	442

を表すツリーの例を示す。

【0107】図5は、項目として上記の同じ文字列を持つ、リスト形式の辞書を示す。リスト形式では、同一の親項目の子供である各項目が、親中の子ポインタによって指定される順次リストに配置される。リスト中の各子項目は、その親の文字列を、辞書で表されるツリー中の異なる文字列経路に拡張する。すなわち、同一の子リスト中の各子は、共通の親文字列からの分岐文字列を提供する。

【0108】辞書内の項目は、子ポインタ・フィールドを持つことができる。子ポインタがない場合、この項目中の文字はある文字列の最後の文字である。子ポインタがある場合、その項目は親であり、そのポインタが、その親の子供である項目のリストを位置指定する。子リスト中の各項目は、その親から異なる文字列を継続する拡張文字 (EC) を含む。

【0109】図6に、図5の子リストを含む圧縮用辞書をさらに展開したものを示す。これらの子リストは、辞書内の順次位置に配置される。同一の親ECを持つ子項目は、辞書内の順次位置にあり、「子リスト」と呼ばれる。辞書内に異なる子リストを連続して配置する必要はない。

【0110】展開用辞書の一般操作—図7：図7に、図

辞書は、圧縮と展開の両方に使用される。この辞書内では、子ポインタ (C PTR) は圧縮だけに使用され、先祖ポインタ (P PTR) は展開だけに使用される。したがって、どの辞書項目も、圧縮と展開の両方をサポートするために C PTR フィールドと P PTR フィールドの両方を有する。P PTR は、索引記号によって位置指定された項目から逆方向に文字を追跡させ、それによって文字列の最後の文字を見つけることにより、ある索引記号用の未圧縮文字列中の文字を再構築することができる。P PTR は、その文字列の以前の文字を持つ以前の項目を位置指定し、前の項目はさらに前の項目の P PTR を含むことができ、以下同様にして、P PTR 値がないことによって、その文字列用の最初の辞書項目が見つかったことが示されるまで続けられる。

【0111】たとえば、記号3は、EC (A) と P PTR 値0を含む、辞書内の項目3を位置指定する。P PTR 値0は、位置3で見つかった EC の前の位置にある、同一の文字列中の別の文字である別の EC (A) を持つ、辞書内の最初の項目を位置指定する。したがって、未圧縮文字列は、このように文字 AA によって再構築される。

【0112】P PTR が0ないし255の範囲の値を持つとき、これがアルファベット項目を指定していることが知られる。この場合、このアルファベット項目は、それ自体の EC を含まないのでアクセスされない。その代わりに、P PTR の値が単に、そのアルファベット項目によって表される EC として使用される。

【0113】展開操作の拡張—図8：図8は、任意の辞書項目中の最大3文字を含む EC フィールドを示す。これらの文字は、当該の項目によって表される少なくとも1つの TEC を含む。この TEC の前には、その左側に文字列内の最大2つの先祖 EC (PEC) がくることがある。このため、1つの辞書項目だけにアクセスすることによって、3つ以下の文字から成る文字列を表すすべての索引記号を展開することができる。この1つの辞書項目は、索引付きの位置にある項目である。文字列中の先祖 EC を位置指定するために P PTR 値が必要となるのは、3つを上回る EC を持つ項目だけである。P PTR で位置指定される項目が P PTR 値を含まないとき、その文字列の展開操作が終了する。

【0114】たとえば、索引記号9は、B の EC と、EC ABC を含む項目7を位置指定する P PTR 値とを含む項目を指定する。ABC は第1の EC B の前に連結されて、文字列 ABCB を形成する。項目7は P PTR 値を持たないので、文字列 ABCB は完全な文字列である。P PTR 値がないので、この連鎖されたアクセス操作は終了する。

【0115】圧縮操作の子拡張—図9：本願発明では、

目中に子文字 (CC) として配置する。これによって、現入力文字と一致しない CC をその親項目中にもつ子項目にアクセスする必要がなくなる。

【0116】図9に、各項目が最大3つの子文字を含む (図4の) 圧縮用辞書の例を表す。図9の項目はまた、図8で説明した方式で展開に使用できる最大3つの EC 文字を含むことができる。

【0117】辞書中のある項目用の子文字が3つを上回る場合、それらの子文字を1つまたは複数の兄弟項目または兄弟記述子中に兄弟文字として配置する必要がある。

【0118】たとえば、図6の辞書構造を使用して文字列 AC を圧縮する場合、位置0にアクセスして文字 A と突き合わせてから、A の子を含む位置3および位置4にアクセスすることになる。この操作の後、C が A の子でないと判定される。図9の新規の辞書構造を用いる場合、前述の場合と同様に、位置0にアクセスして、A と一致するかどうか判定する。前述の手順とは対照的に、この項目の CC フィールドからの位置0の辞書項目から、C が A の子でないと決定することができる。このため、項目位置3および4へのアクセスが不要になる。

【0119】したがって、図9では、親項目中の最大3つの子文字を現入力文字と比較して、現入力文字と一致しない CC があるか否か判定することができる。

【0120】C PTR を持たない子がある場合、その子には子文字がなく、現文字列はその子項目へのアクセスで終了する。その辞書位置は、一致文字列に対応する圧縮済みデータである索引記号である。

【0121】圧縮操作の兄弟拡張—図10：図9では、検査中の親項目に一致する CC が含まれる場合にアクセスが必要であるが、親中で CC が一致しない項目へのアクセスは行われない。

【0122】同様に、図10でも、親項目にスペースが不足しているために、親項目に別の子の CC を含めることができないとき、同じ親の別の子の大部分に対するアクセスが不要になっている。図10は、同じ親の別の子の最初の TEC である最大3つの兄弟文字を含めるのに十分なスペースを持つ項目を示す。このようなとき、SC フィールドの兄弟文字で一致しないものがある場合、その SC を表す項目へのアクセスは不要である。

【0123】兄弟項目の位置は、親項目の CC で指定される最後の子項目からの順次位置である。兄弟項目とは、その親項目の子リスト中の別の子項目にすぎない。SC フィールド中のどの兄弟文字とも一致が見られない場合、それらの SC によって指定される兄弟項目はスキップされる (アクセスされない)。

【0124】入力文字列を AX とすると、項目0の CC A、CC B、CC C がそれぞれ一致しない場合、

比較し、Dが一致しない場合、Xを項目6のSC E、SC F、およびSC Gと比較する。

【0125】辞書項目長因子：辞書中の項目の長さにより、単一項目中で発生できる様々なタイプの文字の組合せおよび数が制限される。辞書中の項目の数により、その項目を参照するのに必要なCPTTRフィールドおよびPPTTRフィールドのサイズが決定される。

【0126】単一の項目に1組の文字を含めるのが適切のように思えるが、8バイトなど合理的なサイズの項目ではそのようにできないことがある。項目の状況に依存する項目構造が、高性能操作のための最良の構造である。

【0127】項目の長さに関するもう1つの重要な考慮点は、実行操作のために辞書項目をプロセッサに転送するプロセッサ・キャッシュおよびレジスタ内の取出し可能ユニットのサイズなど、システムの主記憶域および他の取出し可能メモリ階層レベルでアクセスされるユニットのサイズである。

【0128】辞書項目の制御フィールドー図11、12、13、14、15、16：本願発明はまた、圧縮操作の場合は入力文字ストリーム中の現文字列の終りを決定し、展開操作の場合は文字記号の終りを決定するために必要な項目アクセスの数を減らす助けとなる各種フォーマットを定義するための、1つまたは複数の制御フィールドを辞書項目中に提供する。重要な制御フィールドには、次のものが含まれる。

【0129】1. 子ビット(C)ー現項目が、1つまたは複数の子を持ち、子ポインタ(CPTR)を含む。

【0130】2. more-childrenビット(M)ー現項目が、その項目中の子文字(CC)の数よりも多くの子を持つ。

【0131】3. more-siblingビット(S)ー現項目が、子リスト中でその項目に続く兄弟を持つ。

【0132】4. 記号長(SL)ーこの項目を含む辞書文字列中の文字の数。

【0133】項目の制御フィールドでは、C、M、Sはビット標識であり、SLはカウント・フィールドである。

【0134】Cビットは、項目が子供を持つかどうかを示す。Cは、項目が子供を持たないことを示すとき、辞書文字列が現項目で終わることを示している。その場合、その項目中のすべてのECが入力ストリームと一致するときは、文字列の終りを検出するために別の項目にアクセスする必要はない。

【0135】MビットはCビットと併用されて、その項目中に現入力文字と一致するCCがないとき、さらに兄弟項目がアクセスされることを示す。Mビット情報がなければ、子リストの終りがその項目中の最後のCCに対応するかどうかが分からない。

である兄弟項目で使用されて、その項目中に現入力文字と一致するSCがないとき、さらに兄弟項目にアクセスする必要があることを示す。Sビットは、子リスト中の特殊兄弟記述子中でも同じ目的に使用される。

【0137】項目中にCC用のスペースがないとき、Cビットは1になる。この場合、Mビットを検査する必要はない。同様に、兄弟項目中にSC用のスペースがないとき、Sビットは1になる。

【0138】辞書内の最初の256項目については制御フィールドを小さくすることができる。なぜなら、これらの項目には兄弟がなく(Sが不要)、かつ長さが単一文字だけに事前定義されており、明示的なSLフィールドが必要でないからである。これらの256項目は、バイト中の8ビットのあらゆる置換によって決定される、その辞書のアルファベット文字を表す。アルファベット項目中の制御フィールドは、4ビットだけなので、制御数字(CD)と呼ばれる。CDを図11に示す。

【0139】最初の256項目の後の項目については、制御フィールドにSビットおよびSL値も含まれる。SビットおよびSL値は、必要な任意の最大文字列長を示すことが可能な複数のビットとすることができる。ただし、文字列は必ず最初の256項目のうちの1つで始まる。図12に、最初の256項目の後の項目中で使用される8ビットから成るバイトを占める制御フィールドを示す。この制御フィールドを制御バイト(CB)と呼ぶ。

【0140】辞書が最大で4K(4096)項目に制限される場合、子ポインタ(CPTR)フィールドおよび先祖ポインタ(PPTTR)フィールドのサイズはそれぞれ12ビットとすることができる。これは、4ビットのCDまたは8ビットのCBを収めるのに好都合なサイズである。図13に示すように、辞書内の最初の256項目の1つであるアルファベット項目では、4ビットのCDと12ビットのCPTRを2バイトに収めることができ、次のバイトと、それに続くバイトにCCを格納することができる。アルファベット項目中にECを入れる必要はない。

【0141】最初の256項目の後の項目である非アルファベット項目では、8ビットのCBと12ビットのCPTRを3バイトに収めることができる。この場合、3つのバイトの終りに4つの未使用ビットが残る。これを図14に示す。項目の4番目のバイト中に、EC、その次に他のEC、さらにCCまたはSCを入れることができる。CPTRの代わりに、図15に示すように、PPTTRとすることもできる。

【0142】図16に示すように、CB、CPTR、およびPPTTRを4バイト中に配置することができる。

【0143】複数フォーマット辞書項目ー図17：図17は 複数の項目フォーマットの例。F1A、F1、F

マットは、圧縮・展開併用辞書構造で使うことができる。フォーマットF1AおよびF3Aは、アルファベット項目であり、4ビットの制御数字(CD)を格納する。他のフォーマットは、非アルファベット項目であり、8ビットの制御バイト(CB)を格納する。ここでは、フォーマットF1は、F1兄弟記述子(SD)ではなく文字項目を意味するように使用される。

【0144】フォーマットF1AのCD中の子ビット(C)はゼロであり、子がないことを示す。したがって、フォーマットF1Aはすべてゼロを含む。

【0145】フォーマットF1のCB中のCもゼロである。F1には最大7つのEC用のスペースがある。このうち、1番左のいくつかのECは先祖EC(PEC)とすることができる。図17では、PECと真EC(TEC)を区別しておらず、最初のTECと追加のTEC(AEC)も区別していない。

【0146】F1は、親項目中の子ポインタ(CPTR)が指す子リスト内にある。子ポインタは、少なくとも1の記号長(SL)を持つ必要がある(F1AまたはF3AのSLは暗示の1である)。したがって、F1のCB中のSLは2ないし7の値を持つことができる。ECが占めていないF1のスペースは兄弟文字が占めることができるが、説明を簡単にするため、図17ではE1、E2、、、で表したECだけを示している。F1中のECの数は、F1中のSLである。F1中のECフィールドの始めからF1中の1番左のTECまでのバイト・オフセットが、F1の親中のSLである。この最初のTECの左側にあるECはPECである。F1中のTECの数は、F1中のSLからF1の親中のSLを引いた値である。

【0147】展開の際に索引記号がF1を指定するときは、文字記号はF1中のすべてのECであり、これらのECの数は、F1中のSLで示される。PECとTECを区別する必要はない。索引記号を拡張する際に、他の辞書項目にアクセスする必要はない。

【0148】フォーマットF2のCB中のCもゼロである。F2は最大5つのEC用のスペースを持つ。これらのECの中で、1番左のいくつかのECはPECとすることができる。F2は、CBの次に12ビットの先祖ポインタ(PPTR)を含み、その後4つの未使用ビットが続き、最大5つのEC用のスペースが残る。ECが占めないスペースはSCが占めることができる。F2中のECの数は、F2中のSLから、F2中のPPTRで指定される項目中のSLを引いた値である。F2中のECフィールドの始めからF2中の1番左のTECまでのバイト・オフセットは、F2の親中のSLから、F2中のPPTRで指定される項目中のSLを引いた値である。F2中のTECの数は、F2中のSLから、F2の親中のSLを引いた値である。F2の親とF2中のP

でもよい。

【0149】項目中のCビットがゼロの値で、項目が子を持たないことを示すとき、SLが7以下の場合、その項目はF1であり、SLが8以上の場合、その項目はF2である。これは、F1とF2を区別し、したがって項目がPPTRを含むかどうかを識別する方法である。

【0150】図17に、様々な項目の組合せを示す。図17は、F3AまたはF3中のCPTRによってF1を指定できることを示している(すなわち、CPTRで指定される子リストにその項目を入れることができる)。F1は、F4中のCPTRによって指定することもできるが、これは図では示されていない。図では、F3またはF4中のCPTRによってF2を指定できることを示している。また、F2または別のF4中のPPTRによってF4を直接指定できること、およびF2またはF4中のPPTRによってF3を直接指定できることも示している。

【0151】フォーマットF3AのCB中のCは1であり、その項目が子を持ち、CPTRを含むことを示す。アルファベット項目であるF3A中にはECがないので、最大6つの子文字(CC)用のスペースがある。

【0152】フォーマットF3のCB中のCも1である。F3は最大5つのEC用のスペースを持つ。それらのECのうち1番左のいくつかのECはPECとすることができる。F3は、CBの後に12ビットの子ポインタ(CPTR)を含み、その次に4つの未使用ビットが続き、最大5つのEC用のスペースが残る。ECが占めないスペースはCCが占めることができるが、説明を簡単にするため、図17ではE1、E2、、、で表したECだけを示している。F3中のECの数は、F3中のSLである。F3中のECフィールドの始めからF3中の1番左のTECまでのバイト・オフセットは、F3の親中のSLである。この最初のTECの左側にあるECは、PECである。F3中のTECの数は、F3中のSLからF3の親中のSLを引いた値である。

【0153】フォーマットF4のCB中のCも1である。F4は最大4つのEC用のスペースを持つ。それらのECのうち1番左のいくつかのECはPECとすることができる。F4は、CBの後に12ビットCPTRを含み、その次に12ビットのPPTRが続き、最大4つのEC用のスペースが残る。ECが占めないスペースはCCが占めることができる。F4中のECの数は、F4中のSLから、F4中のPPTRで指定される項目中のSLを引いた値である。F4中のECフィールドの始めからF4中の1番左のTECまでのバイト・オフセットは、F4の親中のSLから、F4中のPPTRで指定される項目中のSLを引いた値である。F4中のTECの数は、F4中のSLから、F4の親中のSLを引いた値である。F4の親とF4中のPPTRによって指定さ

【0154】項目中のCビットが1の値でその項目が子を持つことを示すとき、SLが5以下の場合、その項目はF3であり、SLが6以上の場合にはF4である。これは、F3とF4を区別し、したがってその項目がPPTRを含むかどうかを識別する方法である。

【0155】辞書項目フォーマットの使用例—図18、19、20、21、22

図18に、図19に示す文字記号ツリーに対応する辞書を示す。図20にツリー中の項目のフォーマットを示し、図21に項目中の真拡張文字(TEC)を示し、図22に項目中のすべての拡張文字、先祖EC(PEC)、およびTECを示す。図19、21、22では、項目位置をコロンの前に示し、その後文字記号またはECを示す。

【0156】図18では、位置0にある項目は、文字A用のアルファベット項目である。他の255のアルファベット項目は示していないので、次の項目は位置1にある。項目0(位置0にある項目)は、フォーマットF3Aであり、Cビットを含むがMビットを含まず(すなわち、Cビットの値が1、Mビットの値が0)、項目1を子として指定する子ポインタ(CPTR)を含み、バイトC1中に1つの子文字(CC)Bを含む。バイトC2中のBは、バイトB1中のBの複製であり、したがってバイトB1中のBが最後のCCであることを示す。

【0157】位置1にある項目は、フォーマットF3であり、Cビットと記号長(SL)2を含み、先祖EC(PEC)であるAと真EC(TEC)であるBの2つの拡張文字(EC)を含み、CPTR 2およびCC Cを含む。バイトC2中のCは、バイトC1中のCが最後のCCであることを示す。

【0158】位置2にある項目は、CビットおよびSL 4を含み、PECである4つのEC AB、最初のTECであるC、および追加のTEC(AEC)であるDを含み、CPTR 3およびCC Eを含む。Mビットはゼロなので、項目2の子は1つだけであることが分かる。

【0159】位置3にある項目は、Cビット、Mビット、およびSL 5を含み、ABCDがPECでありEがTECであるEC ABCDEを含み、CPTR 4を含む。項目中にCC用のスペースはない。このため、Cビットが少なくとも1つの子があることを示すので、Mビットは実際には必要でない。

【0160】図18についてさらに説明する。ここでは、すべての詳細は説明せず、図の重要部分だけを説明する。他の詳細は自明である。

【0161】位置4にあるF4項目は、Cビット、CPTR、先祖ポインタ(PPTR)、TEC F、2つのCCであるGおよびXを含み、子リストが位置6から始まることを示す。F4項目はまた、その後の位置5にロ

(SC)はない。項目5中のTECを検査して、項目5に一致が見られるかどうかを判定する必要がある。

【0162】位置5にあるF1項目は、項目4の兄弟である。このF1項目は、PEC ABCDEおよびTEC XYを含む。この項目は、子がなく、SLが7にすぎないので、PPTRを含む項目ではなく、完全な文字記号を含むF1項目とすることができる。

【0163】項目6は、項目4の最初の子である。項目4が2つのCCを含むので、項目7は項目6の兄弟であることが分かり、項目6がSビットを含む必要はない。

【0164】項目7は、項目4の第2の子である。これは、図17の説明で述べたが、図中には示さなかった、F1がF4の子である場合である。

【0165】項目8は、PEC FGおよびTEC Hを含み、項目8を指定する索引記号の展開時に、項目3中のECを項目8中のECの左側に配置すべきことを示すPPTRを含む。

【0166】子文字および兄弟文字の操作—図23および24

図23に、CCおよびSCの位置を使用して親の子リスト中で対応する子供をどのように位置指定するかを示す。図24に、図23に示した項目を含むツリーを示す。

【0167】図23で1番上のF3項目が、対象となる親である。このF3項目は、PECAおよびTEC Bを含み、CC ABCを含む。また、Mビットを含み、ABCの子供よりも多くの子供があることを示す。

【0168】親のCC Aに一致が見られる場合、Aが最初のCCであるので、親中のCPTRに増分0が加算されて、対応する子であるAという子の索引が形成される。A CCには一致が見られないが、B CCに一致が見られる場合、CPTRに増分1が加算され、以下同様である。親中のどのCCにも一致が見られない場合、CPTRに増分3が加算されて、親中のCCに対応する索引の後に最初の子の索引が形成される。この最初の子は、PEC ABおよびTEC Dを含み、SビットおよびSC Eを含む。したがって、親のCCに一致が見られないと、オフセット索引3にある子を取り出され、そのTEC Dでの突合せが行われ、それが失敗した場合は、SC Eでの突合せが行われる。この項目中のSCリストの終りは、バイトS2中の複製Eによって示される。

【0169】オフセット索引3にある項目中のSC Eに一致が見られる場合、そのSCが第1のSCであるので、現項目の索引に増分1が加算されて、SCに対応する兄弟の索引が形成される。これは、増分0が最初のCCに対応する、親中のCCの場合と異なる。この違いの理由は、最初の子の索引であるCPTRにはCCの増分が加算されるが、現項目の索引にはSCの増分が加算さ

【0170】オフセット索引0およびオフセット索引4にある項目中のMビットは、それらの項目中のCCで示される子供よりも少なくとも1つ多い子を示す。図24のツリーでは、それらの追加の子供はTEC Zを持つものと仮定される。

【0171】特殊兄弟記述子項目—図25および26：図25に、親が多数の子供を持ち、おそらく親とその子供に多数のECがあるとき、および子供が子供を持ち、直接の子供がSCではなくCCを含むときに使用される特殊兄弟記述子(SD)を示す。このようなときには、親中のCC用のスペースおよび直接の子供中のSC用のスペースが余りまたはまったくなくなるので、入力文字列の次の文字を子供中の最初のTECと比較して、一致を見つけるために、直接の子供に何度もアクセスを行わなければならない。親の子リスト中に、子リスト中の後続の子供に対応するSCを含むSDを配置すると、このように何度もアクセスを行う必要がなくなる。次いで入力文字列中の次の文字をSD中のSCと比較し、SD中の対応するSCに一致が見られない場合、子リスト中の後続の子供をスキップする。SDについて議論する際、子リスト中の、SDではない項目を文字項目(CE)と呼ぶ。SDがないとき、子リスト中のすべての項目はCEである。

【0172】図26に、その対応する辞書項目が図25の項目を含む、ツリーを示す。

【0173】図25の1番上の項目は、PEC AB、TEC C、ならびにCC AおよびCC Bを含む親である。入力文字列と親のEC ABCの間ですでに一致が見つかったものとする、入力文字列中の次の文字とCC AまたはCC Bの間に一致が見られる場合、突合せプロセスを続行するため、親中のCPTRにそれぞれ増分0または増分1が加算されて、親の対応する子の索引が形成される。CC AとCC Bのどちらにも一致が見られない場合は、CPTRに増分2が加算されて、SDの索引が形成される。親中のMビットは1なので、CCAおよびCC Bに対応する子供以外に、その親には別の子があることが知られる。

【0174】SDの索引が形成されるとき、子リスト中で指定された項目がCEであるかSDであるかは分からない。この項目にアクセスするとき、その制御バイト(CB)中の区別コードがその項目をSDとして識別する。たとえば、CビットがゼロであるがMビットが1であるケースを使用して、ある項目をCEではなくSDとして識別することができる。その項目がCEである場合、突合せプロセスを続行するために、入力文字列中の次の文字がCE中の最初のTECと比較される。その項目はSDなので、入力文字列中の次の文字が、最初のTECではなくSD中のSC CDEFと比較される。SDのバイトS5中のFは、バイトS4中のFの複製であ

を示す。SCCに一致が見られる場合、SDの索引に増分1が加算されて、対応する子の索引が形成される。次に、この対応する子がアクセスされて、突合せプロセスが続行される。SCCには一致が見られないが、SCDには一致が見られる場合、SDの索引に増分2が加算されて、対応する子の索引が形成される。

【0175】図示したように、SDは、4つだけでなく7つのSCを含むことができる。さらに、その場合は7つのSCに対応する数よりはるかに多くの子供がその親にある可能性がある。この場合、SD中のSビットは1となり、SD中のどのSCにも一致が見られない場合、突合せを続行するために7番目の対応する子の後の別の項目にアクセスする必要があることを示す。この次の項目は、CEまたは別のSDとすることができる。

【0176】別個の静的辞書構造

好ましい実施例では、異なる構造を持つ2つの別々の辞書を使用する。一方の辞書は圧縮用であり、もう一方は展開用である。静的辞書の2重辞書構造を提供するのが最善である。というのは、適応(動的)辞書更新操作に2つの異なる辞書の再生成が必要であり、これは単一の適応辞書の更新よりも難しいからである。

【0177】圧縮と展開の両方に単一の静的辞書を使う場合に比べて、別々の静的辞書を使う方が優れた性能を提供することが分かっている。なぜなら、単一の併用辞書は圧縮と展開の両方に最適化することはできず、その性能が、圧縮と展開のどちらか一方に片寄るからである。単一の辞書の効率、別々の辞書の効率に及ばない。

【0178】好ましい実施例では、圧縮用辞書を、展開プロセスには留意せずに圧縮プロセスについて最適化し、展開用辞書を、圧縮プロセスには留意せずに展開プロセスについて最適化する。この最適化は、図29、30、31に示すように、本願発明による別個の辞書内の項目の構造によって行われる。

【0179】圧縮呼出し命令—図27および28：本願発明の好ましい実施例では、圧縮呼出し命令という命令を提供する。この命令を図27に示す。圧縮呼出し(CMPSC)命令は、この命令が使用する汎用レジスタ1(GR1)中のビットによる決定に応じて圧縮または展開を実行する。CMPSCは、圧縮を実行するとき、図1の概要に示すように実行する。CMPSCは、展開を実行するとき、図2の概要に示すように実行する。

【0180】図27に示すように、CMPSCは4ビットのR1フィールドと4ビットのR2フィールドを持つ。これらのフィールドはそれぞれ、偶数/奇数汎用レジスタ対を指定する数を含む(フィールド中の数によって指定される偶数番号のレジスタとそれより1つ小さな奇数番号のレジスタ)。

【0181】図28に、CMPSCが使用するレジス

+1と、やはり暗示的に指定されるGR1を示す。レジスタR1およびR1+1はそれぞれ、宛先オペランドの31ビット・アドレスおよび32ビットの符号なし2進長を含む。宛先オペランドとは、CMPSCが圧縮時にはそこに索引記号を置き、展開時には文字を置く場所である。レジスタR2およびR2+1は、CMPSCが圧縮時にはそこから文字を取りだし、展開時には索引記号を取り出す発信元オペランドのアドレスおよび長さを含む。CMPSCは、処理を完了するために、レジスタR1およびR2中のアドレスを増分し、レジスタR1+1およびR2+1中の長さを減分して、各オペランド中で処理されたデータの量を反映するようにして、新規の開始アドレスおよびオペランドの残りの長さを指定する。

【0182】レジスタGR1は、圧縮済みデータの単位である、索引記号中のビット数と、圧縮用辞書または展開用辞書内の対応する項目の数とを指定する3ビットのフィールドISSを含む。レジスタGR1はまた、圧縮を行うかそれとも展開を行うかを指定するビットも含む。これらのフィールドについての詳細は、図28から自明である。最大13ビットの索引記号サイズが可能であり、これは、前述の単一辞書における12ビットの限界に比べて改善である。索引記号サイズを大きくすれば、辞書が大規模になり、辞書にさらに様々なZL文字列を含めることができるので、圧縮が向上する。

【0183】レジスタGR1は、4K(4096)バイト境界上の辞書の位置を指定する31ビット・アドレスの上位ビットを含む。これは、圧縮操作時に別個の圧縮用辞書であり、あるいは展開操作時に別個の展開用辞書である。

【0184】レジスタGR1はまた、3ビットの圧縮済みデータ・ビット番号(CBN)フィールドを含む。圧縮操作の開始時に、CBNは、圧縮済みデータの第1ビットを配置すべき、R1で指定されるバイト中のビット位置を指定する。この圧縮操作の完了時に、R1が更新されて、圧縮済みデータのビットをまだ含んでいない少なくとも1つのビット位置を含む宛先オペランド中の第1バイトのアドレスを含むようになり、CBNも更新されて、圧縮済みデータを含まないビット位置のうち1番左のビット位置の番号を含むようになる。展開操作の開始時に、CBNが、レジスタR2によって指定されたバイト中で処理すべき圧縮済みデータの第1ビットを指定する。この展開操作の完了時に、レジスタR2が更新されて、処理された圧縮済みデータの最後のビットの後の第1のビットを含む、発信元オペランド中のバイトのアドレスを含むようになり、CBNも更新されて、その第1ビットの番号を含むようになる。ここで説明したレジスタの更新と、レジスタR1+1およびR2+2中の長さの更新は、CMPSC実行の完了時のみならず、たと

割り込まれる場合にも行われる。

【0185】CMPSC実行の完了時に、プログラム状況ワード(PSW、ESA/390の標準部分)中の条件コードが、完了の理由を示すように設定される。条件コードの2つの可能な値を図27に示す。CC0が設定される場合、命令は、操作に応じて、発信元オペランド全体を圧縮または展開したので、所期のとおりの処理を実行し終えている。CC1が設定される場合、宛先オペランド中に処理の出力を受け取るためのスペースがなくなってしまったので、命令は発信元オペランド全体の処理を終えていない。

【0186】別個の圧縮用辞書における項目フォーマット図29

図29に示すように、好ましい実施例の圧縮用辞書内には、文字項目のフォーマット3つと、兄弟記述子フォーマット1つがある。図29には、前述し、図32について説明する際に述べる、代替フォーマット0およびフォーマット1の兄弟記述子は示していない。

【0187】別個の圧縮用辞書の形成には、単一圧縮および展開辞書に関して述べた概念および用語を多数使用する。前述の説明を理解するには、次の説明が必要である。

【0188】単一辞書の場合と同様に、別個の圧縮用辞書内の項目は長さ8バイトである。

【0189】3つの文字項目(CE)はそれぞれ、項目中のCCの数のカウントを含む3ビットの子カウント(CCT)フィールドで始まる。CCTは、単一辞書ではCE中に存在すると記述された子ビット(C)に置き換わる。CCTフィールドの値は、CMPSC命令用の容易な方法で3つの項目フォーマットを区別する。CCTが0のとき、フォーマットは、CPTRやCCを含まず、項目中のAECのカウントを内容とする3ビットのACTだけを含む、C0であることが知られる。AECのカウントは0ないし4となることができる。

【0190】別個の圧縮用辞書では、CEが、そのCEによって新たに表される第1のECである第1の真EC(TEC)を含むことはない。この第1TECは必ず、その項目の親中のCCとして、または親の下にある子リスト中の兄弟記述子(SD)中のSCとしてのみ出現する。CEはまた先行EC(PEC)を含むこともない。PECが有効となるのは展開用辞書内だけだからである。しかし、CEは、その項目によって表され、その第1TECの後に続くTECを含むことができる。これら後続のTECを、前述どおり追加EC(AEC)と呼ぶ。

【0191】フォーマットC0 CEは、0ないし4個のAECを含むことができるので、1つないし5個のTECを表すことができる。

【0192】フォーマットC1 CEは、CCTが1で

み、フォーマットC0と同様に、0ないし4となることが
できるACTを含む。

【0193】別個の圧縮用辞書の好ましい実施例は、単
一辞書に関して説明しなかった新規の構成、すなわち子
検査ビットを含む。別個の圧縮用辞書では、親CEは、
CE中の各CC用の子検査ビット(X)を含む。CCに
一致が見られる場合、関連するXが、0であるかそれと
も1であるか試験する。Xが0の場合、CCに対応する
子項目にアクセスしても意味がないことがただちに分か
る。したがって、現在比較されている入力文字列につい
て、辞書中の最長の一致ZL文字列が見つかったことが
分かる。Xが1の場合、突合せプロセスを続行するため
に一致CCに対応する子にアクセスする必要があることが
分かる。子のXは、子が1つまたは複数のAECを持
つ場合に1となる。なぜなら、これらのAECが入力文
字列と一致しないと、子に一致が見られるようになら
ないからである。子のXは、子がそれ自体の子を持つ場
合も1である。なぜなら、この場合は、入力文字列中の後
続の文字を子のCCまたは子の下にあるSD中のSCと
突合せを試みることによって、突合せプロセスを続行で
きるからである。単一辞書中には子検査ビットは存在し
ない。なぜなら、それらのビットが必要とするスペース
を、単一辞書中の他の情報に使用した方が価値があるか
らである。

【0194】フォーマットC1 CEは、1つのCCを
含むので、1つのXも含む。このXは、CEのビット位
置3にある。

【0195】C1 CEのCCは、CE中の最後のAE
Cの後に続く。AECがない場合、CCはビット位置2
4から始まる。

【0196】フォーマットCG1 CEは1より大きな
CCTを持つ。このフォーマットは、CPTRと幾つか
のCCを含む。また、0または1つのAECを含む。含
むことのできるAECは最大で1つなので、CG1 C
E中のACTフィールドはDという単一ビットに還元さ
れる。Dとは倍文字項目を意味する。Dが1のとき、別
の項目中のCCまたはSCであるCEの第1TECと、
CE中の1つのAECとによって、CEは2つのTEC
を表す。CE中のCCTは、Dが0の場合は2ないし5
となることができ、Dが1の場合は2ないし4となること
ができる。

【0197】フォーマットCG1 CEはまた、その各
CC用のXビットを含み、more-childrenビット(M)
を含む。CE中のCCがCEのすべての子に対応するの
に十分である場合、Mビットは0である。CEの子がC
Cの子よりも多い場合、Mビットは1となって子が多い
ことを示す。Mが1のとき、子リスト中で、CE中のC
Cに対応する最後の子の次にSDが続く。

【0198】兄弟記述子(SD)項目は、SD中のSC

んでいる。このSCの数は1ないし6となることができ
る。SDはまた、その各SC用の子検査ビット(Y)を
含み、more-siblingsビット(S)を含む。図32の理
解を助けるために、SD中の子検査ビットにはXではな
く文字Yを使用する。XがCC用に使われるのと同様
に、YはSC用に使われる。

【0199】圧縮プロセス—図30および31

本願発明の好ましい実施例では、図30および31に示
す一般プロセスを使用して、図29で定義した項目を持
つ構造の別個の圧縮用辞書で入力文字列を圧縮する。

【0200】図30および31では、プロセス内の他の
点から制御が渡されるプロセス内の点を括弧付きの番号
で表し、この説明で参照するために、プロセスのいくつ
かのステップには括弧のない番号を付けてある。プロセ
スは、図30の上端の(1)から開始する。

【0201】(1)で、汎用レジスタR2+1中にある
発信元オペランドの長さが、少なくとも1であるかどう
か試験する。0である場合、CC0がセットされ、実行
は終了する。

【0202】(2)で、汎用レジスタR1+1中にある
宛先オペランドの長さ、汎用レジスタ1中にある圧縮
済みデータ・ビット番号(CBN)を、宛先オペランド
に少なくとも1つの索引記号用のスペースが含まれてい
るかどうかが試験する。試験の方法については、(9)の
操作に関連して説明する。そのスペースが含まれていな
い場合、CC1がセットされ、実行は終了する。図30
および31では、索引記号を単に索引と呼んでいる。

【0203】ステップ41で、汎用レジスタR2によ
ってアドレスされる文字を、汎用レジスタ1中のアドレ
スによってアドレスされる、圧縮用辞書中のアルファベッ
ト項目の索引として使用する。この項目を親項目と呼
ぶ。R2中のアドレスに1が加算され、R2+1中の長
さから1が減算される。

【0204】(3)で、親項目中の子文字(CC)のカ
ウント(CCT)が0であるかどうか試験する。0であ
る場合、制御は(9)に移り、(9)で、親の索引が、
R1によってアドレスされるバイト中に、CBNで指定
されるビット位置から順に記憶されていく。汎用レジス
タ0中の索引記号サイズ(ISS)に応じて9ビット、
10ビット、11ビット、12ビット、または13ビッ
トである索引記号の長さがCBNに加算され、CBNか
らの桁上げが、R1中のアドレスに加算され、R1+1
中の長さから減算される。たとえば、CBNが最初5で
あり、索引記号サイズが13ビットである場合、5と1
3の和は18であり、CBNが2ビットにセットされ、
2バイトがR1に加算され、R1+1から減算される。
制御は(1)に移る。

【0205】親におけるCCTが0でないときは(4)
に達し、(4)で、発信元オペランドが(1)の場合と

まない場合、制御はステップ42に移り、ステップ42で、親の索引がR1、R1+1に記憶され、(9)の場合と同様にCBNが更新され、CC0がセットされ、実行が終了する。

【0206】ステップ43で、発信元オペランド中の次の文字を、親のCCと1番左のCCから順に比較するループを開始する。一致が見られる場合、制御はステップ44に移る。一致がない場合は、(5)で、親中のCCTが別のCCを示しているかどうか試験し、別のCCがある場合はループを繰り返す。

【0207】等しいCCがあるときはステップ44に達するが、ステップ44で、子の索引を、親中の子ポインタ(CPTR)と、等しいCCの番号を加えた値に等しく設定する。この場合、第1のCCの番号は0、次のCCの番号は1となり、以下同様である。

【0208】ステップ45で、親中にあり、等しいCCに対応する子検査ビット(X)が1であるかどうか試験する。0である場合、子が追加EC(AEC)を含まず子を持たないことが分かるので、子の突合せが完了し、可能な突合せはそれが最後であることが分かる。したがって、ステップ46で、宛先オペランド中に子の索引が記憶され、CBNを含むレジスタが更新され、制御が(1)に移る。

【0209】ステップ45でXが1であることが分かった場合は、ステップ47で、子にアクセスし、子中のAECカウント(ACTまたはD)が0であるかどうか試験する。0の場合、子に一致が見られることが分かるので、ステップ48でその子を親と呼び、R2およびR2+1が更新されて1文字進められ、制御が(4)に移る。

【0210】ステップ47で、子中にAECがあることが分かった場合は、ステップ49で、発信元オペランド中の次の文字を、子中にあるだけのAECと比較する準備をする。ステップ50(図31)でまず、発信元オペランドに多数の文字が残っているかどうか試験する。それほど多くの文字が残っていない場合、子の突合せを行うことができないので、制御は(5)に移り、等しいCCがあるかどうか試験するループを続行する。CCは相互に等しくなることができることに留意されたい。この点は、同一の文字が繰り返される、多数の異なる長さの文字列を圧縮する際に好都合である。

【0211】発信元オペランドに文字が十分残っている場合、ステップ51で、それらの文字を子中のAECと比較する。一致が見られる場合、ステップ52で、その子が親になり、発信元レジスタが1にAECの数を加えた分だけ進められ、制御が(3)に移る。一致がない場合、制御が(5)に移り、等しいCCがあるかどうか試験するループを続行する。

【0212】すべてのCCを試験して、子に一致が見つ

ップ53で、親のmore-childrenビット(M)が1であるかどうか試験する。0である場合、親に関する可能な突合せはそれが最後であることが分かるので、制御は(8)に移り、そこで、親の索引が宛先オペランドに記憶され、(9)の場合と同様に宛先レジスタが更新され、制御はさらに(2)に移る。

【0213】ステップ53でMが1である場合、子リスト中に、親のCCに対応する最後の子の次に兄弟記述子(SD)があることが分かる。ステップ54で、親中のCPTRとCCTを加えることによってSDの索引を作成する。

【0214】(7)で、先に親中のCCと比較したが子に一致がなかった、発信元オペランド中の次の文字を、今度はSD中の兄弟文字(SC)と1番左のSCから順に比較していくループを開始する。一致が見られる場合、制御はステップ55に移る。一致がない場合は、ステップ56で、SD中の兄弟文字カウント(SCT)が別のSCを示しているかどうか試験し、別のSCがある場合はループを繰り返す。

【0215】一致がなく別のSCがない場合、ステップ57で、SD中のmore-siblingsビット(S)が1であるかどうか試験する。1である場合、ステップ58で、現SDの索引に現SD中のSCT+1を加算することによって次のSDの索引を作成し、制御が(7)に移り、次のSD中のSCを使用して突合せプロセスを続行する。現SD中のSが0の場合、制御は(8)に移る。

【0216】等しいSCがあるときはステップ55に達するが、ステップ55で、子の索引を、SDの索引に、等しいSCの番号を加えた値に等しく設定する。ここで、第1のCCの番号が1、次のCCの番号が2となり、以下同様である。

【0217】ステップ55より下の諸ステップは、ステップ44より下の諸ステップと同様である。ただし、ステップ59およびステップ60の結果がN0の場合は(5)の代わりに(8)に制御が移る。したがって、SCに一致が見られたが、次の発信元文字がAECに一致しなかった場合、別のSCで一致を見つける試みは行われない。この好ましい実施例では、SCを相互に等しくするのはむだであると規定されていることに留意されたい。これらの等しいSCのうち2番目以降のものは発信元文字と比較されないからである。そうしても、同一の文字が繰り返される、多数の異なる長さの文字列の圧縮効果は下がらないことが分かっており、またそうすると親に関する可能な突合せとしてそれが最後のものであるとより迅速に判断できるので、圧縮速度が上がる事が分かっている。

【0218】別個の展開用辞書中の項目フォーマットー図32: 図32に示すように、好ましい実施例の展開用辞書は、先行なし文字項目と先行あり文字項目という2

【0219】どちらの種類の展開辞書文字項目も、3ビットの部分記号長(PSL)フィールドから始まる。このフィールドは、0を含む場合、項目を先行なし項目として識別し、この項目中の完全記号長(CSL)フィールドは、項目中のECの数のカウント(1ないし7)を含む。これらのECは、先祖EC(PEC)と真EC

(TEC)からなることができる。PECとTECを区別する必要はない。先行なし項目を指定する索引記号または先祖ポインタ(PPTR)を処理すると、その項目中のすべてのECが、宛先オペランド中に次に利用可能な位置から順に配置されていく。次に利用可能な位置とは、前の索引記号の展開から得られる最後の文字を受け取った位置のすぐ後の位置であり、現索引記号が展開すべき最初の索引記号である場合は、宛先オペランドの始めとなる。

【0220】PSLは、0でない場合、その項目を先行あり項目として識別する。その場合、PSLは1ないし5となり、項目中のECの数を示す。その項目は、CSLではなくPPTRを含み、1バイトのオフセット(OFS)フィールドを含む。その項目が索引記号またはPPTRで指定されるとき、項目中のECが、宛先オペランド中に、そのアドレスが次に利用可能な位置のアドレスとOFSの和である位置から順に配置されていく。次に、項目中のPPTRを使用して、先行項目にアクセスする。この先行項目は、別の先行あり項目または先行なし項目とすることができる。

【0221】いくつかの先行あり項目へのアクセス、次いで先行なし項目へのアクセスによって索引記号を展開するとき、その辞書が論理的に正しいものとする、各エントリ中のECが、先行項目のために配置されたECに隣接しその左側にある宛先オペランド中に配置される。しかし、辞書が論理的に正しくないかどうかの検査は行われない。辞書が正しくない場合、宛先オペランド中にギャップができたり、文字が重なることがある。

【0222】索引記号が先行なし項目を指定するとき、その項目中のCSLは、次の索引記号を展開するために、宛先オペランド中の次に利用可能な位置が前に進められる量である。索引記号が先行あり項目を指定するとき、その第1の先行あり項目中のPSLとOFSの和が、次の索引記号のために次に利用可能な位置が前に進められる量である。以後の先行あり項目中のPSLおよびOFSと、最後の先行なし項目中のCSLは、この前進量の計算には関与しない。

【0223】第1の(または任意の)先行あり項目は最大5のPSLと最大255のOFSを含むことができるので、索引記号に対応する文字記号の最大長は260文字である。

【0224】展開プロセス—図33：図30および31の場合と同様、図32では、索引記号と、宛先オペランド

ているものとする。

【0225】(1)で、R2+1中の発信元オペランド長とCBNを、発信元オペランドに少なくとももう1つの索引記号が含まれるかどうか試験する。含まれていない場合、CC0がセットされ、実行は終了する。

【0226】ステップ71で、次の索引記号が256より小さな値を持つかどうか試験する。256未満の値を持つ場合、この索引記号がアルファベット項目を指定していることが分かり、制御はステップ72に移る。

【0227】ステップ72で、R1+1中の宛先オペランド長を、宛先オペランドに少なくとも1つの文字位置が残っているかどうか試験する。残っていない場合、CC1がセットされ、実行は終了する。残っている場合、ステップ73で、索引記号の1番右の8ビットを、R1で指定される位置に展開済みデータの文字として配置する。1がR1に加算され、R1+1から減算される。R2、R2+1、およびCBNが、図30の制御点(9)に関する説明と同様に更新され、制御は(1)に移る。

【0228】索引記号が255を上回るときステップ74に達し、ステップ74で、索引記号を使用して辞書項目にアクセスする。その後、この辞書項目は現項目と呼ばれる。

【0229】ステップ75で、現項目中の部分記号長(PSL)が0であるかどうか試験する。0である場合、その項目は先行なし項目であることが分かり、制御はステップ76に移る。

【0230】ステップ76で、現項目から完全記号長(CSL)を取り出し、次いで宛先オペランド中にCSL文字位置が残っているかどうか試験する。残っていない場合、CC1がセットされ、実行は終了する。残っている場合、ステップ77で、その項目からCSL ESを取り出し、それを宛先オペランド中にR1中のアドレスから順に配置していき、レジスタを更新し、制御は(1)に移る。

【0231】PSLが非ゼロであり、先行あり項目を示すとき、ステップ78に達する。この項目は第1の先行あり項目なので、(記号長を表す)SYMLENという変数が、項目中のPSLおよびOFSの和で設定される。

【0232】ステップ79で、宛先オペランド中にSYMLEN文字位置が残っているかどうか試験する。残っていない場合、CC1がセットされ、実行は終了する。残っている場合、ステップ80で、その項目からPSL ECを取り出し、それを宛先オペランド中にR1中のアドレスとOFSを加えた位置から順に配置していく。ステップ80では次に、項目中の先祖ポインタ(PPTR)を使用して、PPTRが指定する項目にアクセスし、その後その新規にアクセスされた項目が、この説明では現項目と呼ばれる。

が0であるかどうか試験する。0である場合、ステップ82で項目からCSL ECを取り出し、それをR1中のアドレスから順に配置していく。ステップ82では次に、レジスタを更新し、制御が(1)に移る。具体的には、ステップ82でR1にSYMLENが加算され、R1+1からSYMLENが減算される。

【0234】ステップ83で、項目からPSL ECを取り出し、それをR1中のアドレスとOFSTを加えた位置から順に配置していく。SYMLENが変更されないことに留意されたい。ステップ83では次に、項目中のPPTRを使用して、新規の現項目にアクセスした後、制御がステップ81に移る。

【0235】ステップ80およびステップ83に関する注で示したように、PPTRが256より小さな場合、指定された項目を先行なし項目とみなすことができ、その項目にアクセスすることによってECを取り出す代わりに、PPTRの1番右の8ビットがその項目中のECとして使用される。

【0236】繰返し文字を圧縮するための辞書—図34：図34は、別個の辞書に関する圧縮プロセスおよび展開プロセスをさらに理解するうえで役立ち、また項目に複数のAECを含めることができ、親中のCCを同一にすることができるという本願発明の規定が、同一の文字が繰返される、多数の異なる長さの文字列を圧縮する際にどのように有益かを示している。図34では例として文字Aが選択されている。8つのゼロ・ビットと空白文字から構成される文字の方が重要であるが、印刷が不可能であり、例示するのが容易でない。

【0237】図34に、圧縮用辞書内の一定の索引位置と、展開用辞書内の同一の索引位置を表したものである。文字Aは10進値が193なので、索引位置193におけるAのアルファベット項目が示されている。他のアルファベット項目は示されていない。Aの子孫は索引位置256から示されている。索引位置256は、アルファベット項目の後の最初の位置である。

【0238】圧縮用辞書に示されている項目にはCCより多くの子を持つものがないので、図34にはMビットは示されていない。

【0239】以下は、圧縮用辞書の項目を理解するための説明である。

【0240】項目193は、5であるCCT、11110である5つの対応するXビット、0であるD、256であるCPTR、およびそれぞれ値Aを持つ5つのCCを有するものとして示されている。

【0241】項目256は、1であるCCT、1である1つの対応するXビット、4であるACT、261であるCPTR、それぞれ値Aを持つ4つのAEC、および値Aを持つ1つのCCを有するものとして示されている。

ACT、それぞれ値Aを持つ3つのAECを有し、Xビット、CPTR、CCを有さないものとして示されている。

【0243】項目193、256、257についての以上の説明から、圧縮用辞書の残りの項目は自明である。

【0244】圧縮用辞書の後の2つの列は辞書の一部ではない。これらの列には、辞書項目に関する情報だけが含まれる。圧縮辞書項目の後の最初の列の数は、その項目（および同一の番号の展開辞書項目）で表されるAの数であり、第2の列の数は、圧縮辞書項目の突合せを行うのに必要な記憶域参照の回数である。たとえば、入力文字列がAAAAABである場合、突合せプロセスは次の手順から構成される。(1)文字列中に第1のAがあるため項目193を参照する。(2)文字列中の第2のAを項目193中の第1のCC Aと突き合わせ、次いで項目256を参照し、文字列中に残っているAAABが項目256中のAEC AAAAと一致しないことを確認する。(3)文字列中の第2のAを項目193中の第2のCC Aと突き合わせ、次いで項目257を参照し、文字列中に残っているAAABのAAAが項目257のAEC AAAと一致することを確認する。(4)項目257中のCCTが0なので、文字列中に現在残っているBに一致が見られる可能性がなく、したがって項目257に関する可能な突合せはそれが最後のものであることを確認する。したがって、5つのAを表す圧縮辞書項目に関するこの最後の可能な突合せを見つける際に、記憶域参照が3回行われた。

【0245】以下は、展開辞書項目を理解するための説明である。

【0246】展開すべき索引記号が270であるものとする。圧縮辞書項目の後の最初の列は、圧縮用辞書または展開用辞書中の項目270が16個のAを表すことを示している。この情報は、本説明を理解する助けとなるものにすぎず、展開プロセスでは必要とされない。

【0247】展開用辞書中の項目270は、4であるPSL、262であるPPTR、それぞれ値Aを持つ4つのEC、12であるOFSTを含み、CSLを含まない。

【0248】項目262は、5であるPSL、261であるPPTR、それぞれ値Aを持つ5つのEC、7であるOFSTを含み、CSLを含まない。

【0249】項目261は、0であるPSL、7であるCSL、それぞれ値Aを持つ7つのECを含み、PPTRとOFSTを含まない。

【0250】索引記号270の展開は次の手順から構成される。(1)項目270を参照し、その項目からAAAAを取り出し、それを宛先オペランド中の次に利用可能な位置からオフセット12の所に置く。(2)項目262を参照し、その項目からAAAAAを取り出し、そ

ット7の所に置く。(3)項目261を参照し、その項目からAAAAAAAAを取り出し、それを宛先オペランド中の次に利用可能な位置に置く。16個のAを圧縮するには8回の記憶域参照が必要であるが、拡張するには3回だけでよいことに留意されたい。

【0251】圧縮用辞書の項目が、同一の文字が繰り返される、多数の異なる長さの文字列が圧縮できる明白な形の構造である場合、Aのアルファベット項目が第2のAを表す単一の子を持ち、その子が第3のAを表す単一の子を持ち、その子が第4のAを表す単一の子を持ち、以下同様となる。したがって、Aの文字列を突き合わせるのに必要な記憶域参照の回数は、文字列中のAの数に等しくなる。

【0252】図34の圧縮辞書項目の新規パターンにより、上述の明白なパターンの場合よりも圧縮時の記憶域参照回数を減らすことができる。実際に、長い文字列の限界では、記憶域参照の回数を67%削減することができる。たとえば、図34では、12個のAを突き合わせるのに4回、18個のAを突き合わせるのに6回、24個のAを突き合わせるのに8回の参照が必要であり、したがって参照の回数はAの数の33%である。

【0253】図34のパターンについて以下に説明する。奇数レベルの親はAECを持たず5つの子を持ち、それらの子供はそれぞれ4つ、3つ、2つ、1つ、0個のAECを持つ。これらの子供は、親よりも5つないし1つ多い文字を表す。偶数レベルの親は、奇数レベルの親の第1の子であり、4つのAECおよび1つの子を持つ。この子は、次の奇数レベルの親である。奇数レベルの親の2番目ないし5番目の子は、子を持たない。図34では、項目193、261、267、273、279が奇数レベルの親であり、項目256、262、268、274が偶数レベルの親である。

【0254】上述のパターン方式では、奇数レベルの親が6つ以上の子を持たないようにすることも、奇数レベルの親の2番目ないし5番目の子が子を持たないようにすることもできない。これらの追加の子はいずれも、同一の繰返し文字以外も含むZL文字列を形成するのに役立つことがある。

【0255】短記号オプション—図35、36、37 図35は、圧縮すべきデータが非常にランダムであるために、その2つ以上の隣接文字のうち辞書中にZL文字列として出現する文字があまりないときに、使用するために短記号オプションをどのように呼び出すことができるかを示す。この状態は、このデータの2つの隣接バイトが多数の異なる値を持つ可能性があるので、データが2進データまたはバック10進データであるときに生じる可能性が高い。

【0256】図36は、短記号オプションを使用するとき、索引記号の左にゼロ・ビットを置く手順を示してい

し13ビットの任意の長さとしてすることができ、その際、長記号の長さはそれぞれ10ビットないし14ビットとなる。しかし、一般に、索引記号長が9ビットまたは10ビットの場合は短記号オプションは使用すべきではない。なぜなら、そうすると、先行ゼロ・ビットによってスペースが大幅に使用され、圧縮度が下がる可能性があるからである。

【0257】短記号オプションを使用するとき、1文字だけから成る文字記号を表す索引記号から長記号が作成されることはない。その代わり、単一の未圧縮文字を短記号と呼び、これをいわゆる短記号文字列中の出力圧縮済みデータ中に置く。短記号文字列は4ビットの短記号ヘッダで始まり、次に1つないし8つの短記号が続く。このヘッダは、長記号と区別するためゼロ・ビットで始まり、その後文字列中の短記号の数を示す3ビットのカウント・フィールドを含む。このカウント・フィールドが0の場合は1つの短記号を示し、1の場合は2つの短記号を示し、以下同様である。

【0258】文字列に1つの短記号だけが含まれるとき、短記号文字列は圧縮度に対してほとんど効果を与えず、ときには悪影響を与える場合もある。短記号文字列は、複数の短記号を含むときに効果が高くなる。たとえば、図37は、3つの短記号から成る短記号文字列が28ビットであることを示している。索引記号サイズが12ビットである場合、3つの索引記号は合計36ビットの長さを持ち、それに対応する3つの長記号は合計39ビットの長さを持つ。

【0259】以上で述べた実施例から抽出することのできる発明には、以下のような態様がある。

【0260】(1) Ziv-Lempel (ZL) 圧縮アルゴリズムを実施して比較的大規模なデータベース内の任意の1つまたは複数の比較的小規模な未圧縮レコードを圧縮することによりレコードの圧縮を改善する方法において、前記データベースに前記ZL圧縮アルゴリズムを使用するコンピュータ・プログラムを適用することにより、圧縮済みレコードを生成する前に、前記データベース内のすべてのZL文字列を辞書文字列として含む静的圧縮辞書を事前に生成しておくステップと、別のコンピュータ・プログラムを実行して、未圧縮レコード中の一連の文字を前記辞書中の前記辞書文字列と突き合わせることにより、前記未圧縮レコード中のレコード文字列を検出するステップと、前記辞書中の辞書文字列と突き合わされるレコード文字列の、前記辞書内での終了位置を表す索引記号を出力して、前記未圧縮レコードに対応する圧縮済みレコードを提供するステップとにより、前記データベース内の未圧縮レコードが変更されているか否かにかかわらず、前記辞書を変更せずに、アクセスされた前記未圧縮レコードから前記圧縮済みレコードを生成するステップとを含むレコード圧縮方法。

装置に前記辞書を記憶するステップと、各辞書項目ごとに、前記コンピュータ・システムの記憶装置からアクセス可能な記憶装置アクセス・ユニットのサイズに等しい固定サイズを選択するステップとをさらに含む(1)記載のレコード圧縮方法。

【0262】(3)前記データベース内の各文字列を、1または複数の辞書項目によって表される辞書文字列として構造化するステップであって、各前記項目は、前記辞書文字列中の1または複数の対応する拡張文字(ESC)が割り当てられ、前記割り当てられたESCは、前記辞書項目中に記録されることもされないこともあり、真拡張文字(TEC)と呼ばれる前記ステップと、辞書文字列中の前記辞書項目を、2つ以上の項目によって表される任意の文字列の第1項目から連鎖または索引付けするステップとを含む(1)記載のレコード圧縮方法。

【0263】(4)子ESCに割り当てられた前記辞書項目にアクセスせずに子文字(CC)からレコード文字列を検出できるとき、前記圧縮済みレコードを生成するプロセスにおける前記辞書の記憶装置へのアクセスを削減するために、任意の辞書項目中で、同一の辞書文字列中の辞書項目に割り当てられたESCの次に、子文字(CC)と呼ばれる、1つまたは複数のESCの複製を作成するステップをさらに含む(3)記載のレコード圧縮方法。

【0264】(5)レコード文字列中の第1の文字を表す辞書項目から辞書に入り、未圧縮レコードから順次得られる後続の各レコード文字またはレコード文字の文字列を同一の辞書文字列中の後続の辞書項目と比較して前記レコード文字列の終りを決定することにより、前記未圧縮レコード中の各レコード文字列を検出するコンピュータ・プログラムを実行し、前記辞書文字列中の最後の前記項目を検出するか、あるいは前記辞書文字列中のESCと一致するレコード文字の後に続く次のレコード文字と一致しない前記文字列中のESCを検出することにより前記レコード文字列の終りを位置指定するステップをさらに含む(3)記載のレコード圧縮方法。

【0265】(6)前記事前生成ステップで辞書が提供されて以降に、前記データベース内の1つまたは複数のレコードが変更され、あるいは前記データベースに追加された、データベース内の未圧縮レコードから、前記静的圧縮辞書を使って、前記辞書をデータベースの変更に適合させずに、レコードを圧縮及び展開するように設計された辞書を使用する場合よりも速い速度で、前記圧縮済みレコードを生成するステップをさらに含む(1)記載のレコード圧縮方法。

【0266】(7)前記未圧縮レコードが変更された場合でも、前記静的圧縮辞書を使って、前記辞書をデータベースの変更に適合させずに、レコードを圧縮及び展開

含む(1)記載のレコード圧縮方法。

【0267】(8)前記適応辞書を使用する場合には実行できない方式でレコードを圧縮するために、前記データベースを任意の順序で(レコードを取り出す順序と同じランダムな順序、またはそれと異なるランダムな順序で、あるいは順次に)走査することによって、前記静的圧縮辞書が事前に生成された場合でも、該辞書を使って、前記データベースからランダムな順序で取り出された前記未圧縮レコードを圧縮するステップをさらに含む(1)記載のレコード圧縮方法。

【0268】(9)前記静的圧縮辞書を、同一の辞書文字列を含む静的展開辞書と関連付けるステップと、レコードを圧縮及び展開するように設計された辞書を使用する場合よりも速い速度でレコードを展開するために、圧縮済みレコード中の索引記号を使って展開辞書にアクセスして、前記索引記号で表される文字列を取り出すことにより、前記圧縮辞書を使って圧縮された圧縮済みレコードを展開するステップとをさらに含む(1)記載のレコード圧縮方法。

【0269】(10)前記静的圧縮辞書と関連付けられた送信装置と、静的展開辞書と関連付けられた受信装置とを有する伝送ネットワークを提供するステップと、辞書を伝送せずに、前記伝送ネットワークを介して前記送信装置から受信装置に圧縮済みレコードを伝送するステップとをさらに含む(9)記載のレコード圧縮方法。

【0270】(11)前記静的圧縮辞書を使って圧縮済みレコードを生成した後、記憶媒体内の前記データベースに各圧縮済みレコードを記憶するステップと、前記記憶媒体内のデータベースから圧縮済みレコードにアクセスするステップと、前記静的展開辞書を使って、レコードを圧縮及び展開するように設計された辞書を使用する場合よりも速い速度で、レコードを展開するステップとをさらに含む(9)記載のレコード圧縮方法。

【0271】(12)それぞれ辞書文字列中に1または複数の先祖ESCを有する子ESCを前記辞書文字列中に有する次の辞書項目を位置指定するために、前記辞書文字列の少なくとも第1の項目に子ポインタを記憶するステップをさらに含む(3)記載のレコード圧縮方法。

【0272】(13)第1の辞書項目で表されるアルファベットESCの値によって示される位置に各辞書文字列の各第1辞書項目を位置指定し、前記各第1辞書項目を前記アルファベット項目として指定し、他のすべての辞書項目を非アルファベット項目として指定するが、各アルファベット項目の位置が、割り当てられたアルファベットESCと関連付けられているために、どの前記アルファベット項目中にもESCを必要としないステップをさらに含む(12)記載のレコード圧縮方法。

【0273】(14)前記静的圧縮辞書を、ESCおよび

法。

【0274】(15) 同一の辞書文字列中の直接の先祖項目中の子ポインタによって前記非アルファベット辞書項目を位置指定するステップをさらに含む(14)記載のレコード圧縮方法。

【0275】(16) 前記構造化ステップが、前記非アルファベット項目中の前記制御フィールドを、前記項目に割り当てられた真ECの数を示すECカウント表示を伴う構造にするステップをさらに含む(14)記載のレコード圧縮方法。

【0276】(17) 前記構造化ステップが、前記アルファベット項目または非アルファベット項目の前記制御フィールドを、親項目に割り当てられたECの後に続く、辞書文字列中の文字である子EC(CC)をいつ前記親項目が含むかを示す子標識を含む親辞書項目(親項目)として構造化し、それによって、前記CCに割り当てられた前記子辞書項目への追加のアクセスなしで、レコード文字列の終りを検出できるようにすることにより、前記CCが前記辞書内のレコード文字列の検出効率を向上できるようにするステップをさらに含む(14)記載のレコード圧縮方法。

【0277】(18) 前記親項目中に含まれるCCの数を示すカウント標識を前記親項目中に置き、各CCが、関連する子項目に割り当てられた第1のECであるか、または関連する子項目に割り当てられた1組の第1ECであることを事前決定しあるいは項目中で指示するステップをさらに含む(17)記載のレコード圧縮方法。

【0278】(19) 1つまたは複数の子辞書項目(子項目)を有する子リストを含む、前記親項目の前記子辞書項目を提供するステップと、前記子リスト中の第1の子項目に対して所定の位置に、前記子リスト中の各前記子項目を位置指定するステップと、前記親項目中の各CCを前記子リスト中の当該の子項目と関連付けられるように割り当てるステップとをさらに含む(18)に記載のレコード圧縮方法。

【0279】(20) 辞書内の前記子リストを前記親項目中の子ポインタ・フィールドによって位置指定するステップをさらに含む(19)記載のレコード圧縮方法。

【0280】(21) 現在検出中のレコード文字列用の索引記号を生成するために、子ポインタと、辞書文字列中のECと突き合わされる次のレコード文字と一致する親項目中の関連CCの位置とから、子辞書項目の位置を算出するステップをさらに含む(20)記載のレコード圧縮方法。

【0281】(22) 親項目の制御フィールド中のmore-children標識をオンにセットして、前記親項目内に含まれる前記CCの数よりも多くの前記子項目が前記親項目の前記子リスト中にあることを示すステップをさらに含む(19)記載のレコード圧縮方法。

り当てられた子項目で一致が見つかった場合、ECはすでに分かっているため、割り当てられたECが親項目のCCであるとき、前記圧縮辞書項目から、割り当てられたECを省略するステップをさらに含む(17)記載のレコード圧縮方法。

【0283】(24) 子検査標識を親項目中のCCと関連付けて、CCが現レコード文字と一致するとき、他の辞書項目にアクセスする必要なく、CCが辞書文字列を終了する(したがって、レコード文字列の終りを検出する)かどうかを示すステップをさらに含む(17)記載のレコード圧縮方法。

【0284】(25) 子が指定CC以外に割り当てられたECを持たず、かつそれ自体の子を持たないことが事前決定されているとき、それ以上突合せプロセスを続行できないので、関連する子項目(子)へのアクセスが必要でないことを示すように、CCと関連する子検査標識を設定するステップ、または子が指定CC以外に割り当てられたECを持ち、あるいはそれ自体の子を持つことが事前決定されているとき、子に真の一致が見られるかどうか、あるいは突合せプロセスが続行できないかどうか分からないので、子へのアクセスが必要であることを示すように、子検査標識を設定するステップをさらに含む(24)記載のレコード圧縮方法。

【0285】(26) more-children標識を含む前記親項目(親)の下にある前記子リスト中の前記子項目

(子)中に、前記親中の利用可能なスペースに親中のCCと区別するために兄弟文字(SC)と呼ばれるその当該の各識別CCを含めることができない、同一の親の子供と関連するCCを置き、各SCが前記同一の親の関連する子に割り当てられた第1のECであるか、または前記同一の親の関連する子に割り当てられた1組の第1ECであることが事前決定され、あるいは項目中で指示されており、それによって、前記SCに割り当てられた前記子辞書項目への追加のアクセスなしで、前記レコード文字列の終りを検出できるようにすることにより、前記SCが前記辞書内の前記レコード文字列の検出効率を向上できるようにするステップと、前記子項目に含まれるSCの数を示すカウント標識を前記子項目中に置くステップと、SCを含む前記子項目中の各SCを、前記子リスト中の当該の子項目と関連付けられるように割り当てるステップとをさらに含む(22)記載のレコード圧縮方法。

【0286】(27) SCを含む子項目を、親中に含まれる最後のCCによって指定される子項目の位置に対して所定の方式で位置指定するステップをさらに含む(26)記載のレコード圧縮方法。

【0287】(28) 現在検出中のレコード文字列用の索引記号を生成するために、SCを含む第1の子項目の位置と、辞書文字列中のECと突き合わされる次のレコ

置とから、第2の子項目の位置を算出するステップをさらに含む(27)記載のレコード圧縮方法。

【0288】(29) SCを含む子項目の制御フィールド中のmore-siblings標識をオンにセットして、子項目が、子項目内に含まれるよりも多くのSCを有することを示すステップと、SCを含む第1の子項目中の最後のSCによって指定される子項目の位置に対して所定の位置に、上記の追加SCを含む別の子項目を置くステップとをさらに含む(27)記載のレコード圧縮方法。

【0289】(30) 同一のSCを使用して、ECに割り当てられた子項目に一致が見つかった場合、ECはすでに分かっているの、同一の子リスト中の別の子項目中の同一のSCによって位置指定された子項目から、割り当てられたECを省略するステップをさらに含む(26)記載のレコード圧縮方法。

【0290】(31) 子検査標識を子項目中のSCと関連付けて、SCが現レコード文字と一致するとき、他の辞書項目にアクセスする必要なしに、SCが辞書文字列を終了する(したがって、レコード文字列の終りを検出する)かどうかを示すステップをさらに含む(26)記載のレコード圧縮方法。

【0291】(32) 子が指定SC以外に割り当てられたECを持たず、かつそれ自体の子を持たないことが事前決定されているとき、それ以上突合せプロセスが続行できないので、関連する子項目(子)へのアクセスが必要でないことを示すように、SCと関連する子検査標識を設定するステップ、または子が指定SC以外の割り当てられたECを持ち、あるいはそれ自体の子を持つことが事前決定されているとき、子に真の一致が見られるかどうか、あるいは突合せプロセスが続行できないかどうか分からないので、子へのアクセスが必要であることを示すように、子検査標識を設定するステップをさらに含む(31)記載のレコード圧縮方法。

【0292】(33) more-children標識を含む親項目(親)の下にある子リスト中に、前記アルファベット項目でも非アルファベット項目でもなく、前記親中の利用可能なスペースに、前記親中のCCと区別するために兄弟文字(SC)と呼ばれるその当該の各識別CCを含めることができない、前記同一の親の子項目(子)と関連するCCを含む、兄弟記述子(SD)と呼ばれる項目を置き、各SCが前記同一の親の関連する子項目(子)に割り当てられた前記第1のECであるか、または前記同一の親の関連する子に割り当てられた1組の前記第1ECであることが事前決定され、あるいは前記SD中で指示されており、それによって、前記SCに割り当てられた前記辞書項目への追加アクセスなしで、レコード文字列の終りを検出できるようにすることにより、前記SCが前記辞書内のレコード文字列の検出効率を向上でき

記SD中の各SCを、前記子リスト中の当該の子項目と関連付けられるように割り当てるステップとをさらに含む(22)記載のレコード圧縮方法。

【0293】(34) 前記SDを、親内に含まれる最後のCCによって指定される子項目の位置に対して所定の方式で位置指定するステップをさらに含む(33)記載のレコード圧縮方法。

【0294】(35) 現在検出中のレコード文字列用の索引記号を生成するために、SCを含むSDの位置と、辞書文字列中のECと突き合わされる次のレコード文字と一致するSD中の関連するSCの位置とから、子項目の位置を算出するステップをさらに含む(34)記載のレコード圧縮方法。

【0295】(36) SDの制御フィールド中のmore-siblings標識をオンにセットして、SDが、SD内に含まれるよりも多くのSCを有することを示すステップと、第1のSD中の最後のSCによって指定される子項目の位置に対して所定の位置に、上記の追加SCを含む別のSDを置くステップとをさらに含む(34)記載のレコード圧縮方法。

【0296】(37) 同一のSCを使用して、ECに割り当てられた子項目に一致が見つかった場合、ECはすでに分かっているの、同一の子リスト中のSD中の同一のSCによって位置指定された子項目から、割り当てられたECを省略するステップをさらに含む(33)記載のレコード圧縮方法。

【0297】(38) 子検査標識をSD中のSCと関連付けて、SCが現レコード文字と一致するとき、他の辞書項目にアクセスする必要なしに、SCが辞書文字列を終了する(したがって、レコード文字列の終りを検出する)かどうかを示すステップをさらに含む(33)記載のレコード圧縮方法。

【0298】(39) 子が指定SC以外に割り当てられたECを持たず、かつそれ自体の子を持たないことが事前決定されているとき、それ以上突合せプロセスが続行できないので、関連する子項目(子)へのアクセスが必要でないことを示すように、SCと関連する子検査標識を設定するステップ、または子が指定SC以外の割り当てられたECを持ち、あるいはそれ自体の子を持つことが事前決定されているとき、子に真の一致が見られるかどうか、あるいは突合せプロセスが続行できないかどうか分からないので、子へのアクセスが必要であることを示すように、子検査標識を設定するステップをさらに含む(38)記載のレコード圧縮方法。

【0299】(40) アルファベット項目または非アルファベット項目が占めることのできる圧縮辞書内だけのスペースにSDを置くステップと、展開辞書内のこのスペースが、アルファベット項目または非アルファベット

含むことができないため、該スペースが普通なら無駄になるので、圧縮辞書内のSDの位置と同一の展開辞書内の位置にもSDを置くステップとをさらに含む(33)記載のレコード圧縮方法。

【0300】(41)辞書項目における終了の指示が、辞書文字列の終りであることを示すとき、辞書項目内のすべてのECがレコード文字と一致するとき、レコード文字列を検出するステップをさらに含む(1)記載のレコード圧縮方法。

【0301】(42)比較的大規模なデータベースの対応する未圧縮レコードを再構築するために、(5)記載の方法によって生成される圧縮済みレコードを展開するための圧縮済みレコード展開方法であって、ZLアルゴリズムを使用するコンピュータ・プログラムを、前記圧縮辞書の構築に使用するデータベースに適用することによって、前記データベースの未圧縮レコードを再構築する前に、静的展開辞書を事前生成しておき、前記圧縮辞書内の辞書文字列に対応する辞書文字列を表す辞書項目を前記展開辞書内で構築するステップと、圧縮済みレコード中にありかつ未圧縮レコード中のレコード文字列を表す各索引記号と関連する1つまたは複数の展開辞書項目を見つけることによって、前記索引記号を検出する別のコンピュータ・プログラムを実行するステップと、見つかった展開辞書項目から各索引記号で表される各レコード文字列用の文字にアクセスするステップと、圧縮済みレコードから入力される各索引記号用の前記アクセスされた文字を出力して、対応する未圧縮レコードを再構築するステップとによって、前記圧縮辞書および前記展開辞書が事前生成された後に変更された未圧縮レコードからデータベース内の圧縮済みレコードが生成されたかどうかにかかわらず、前記展開辞書を変更せずに、入力された圧縮済みレコードから未圧縮レコードを再生成するステップとを含む圧縮済みレコード展開方法。

【0302】(43)コンピュータ・システムの記憶域に辞書を記憶するステップと、コンピュータ・システムの記憶域からアクセス可能な記憶域アクセス・ユニットのサイズと等しい、各辞書項目の固定サイズを選択するステップとをさらに含む(42)記載の圧縮済みレコード展開方法。

【0303】(44)展開中の圧縮済みレコードが、静的展開辞書が事前生成された後に変更された対応する未圧縮レコードを表す場合でも、同一の静的展開辞書を使用してデータベースの圧縮済みレコードを展開し、したがって、静的展開辞書を対応する未圧縮レコードに適合させる必要のないステップをさらに含む(42)記載の圧縮済みレコード展開方法。

【0304】(45)適応辞書を使用しては実行できない方式でレコードを圧縮し展開するために、データベースから任意の順序で(データベースから取り出した順

序で、あるいは順次に)走査することにより、静的展開辞書が事前生成された場合でも、静的展開辞書を使用して、データベースからランダムな順序で取り出された圧縮済みレコードを展開するステップをさらに含む(42)記載の圧縮済みレコード展開方法。

【0305】(46)適応辞書を使用しては実行できない方式でレコードを圧縮しかつ展開するためにデータベースから任意の順序で(データベースから取り出した順序と同じランダムな順序、またはそれと異なるランダムな順序で、あるいは順次に)取り出した未圧縮レコードを圧縮することによって圧縮済みレコードが生成された場合でも、静的展開辞書を使用して、データベースからランダムな順序で取り出した圧縮済みレコードを展開するステップをさらに含む(42)記載の圧縮済みレコード展開方法。

【0306】(47)データベース用に受信された圧縮済みレコードを展開するために、ネットワークの受信端で展開辞書を記憶するステップをさらに含む(42)記載の圧縮済みレコード展開方法。

【0307】(48)ネットワークの受信端で静的展開辞書を使用して圧縮済みレコードを展開し、各未圧縮済みレコードの伝送と共に辞書を伝送することなしに、対応する未圧縮レコードを生成するステップをさらに含む(42)記載の圧縮済みレコード展開方法。

【0308】(49)索引記号で位置指定された前記項目内に含まれるよりも多くの先祖文字が関連する前記辞書文字列中に存在するとき、索引記号で位置指定された前記項目に連鎖された、前記展開辞書内の1つまたは複数の先祖辞書項目を提供するステップと、前記索引記号で位置指定された前記項目中に、先祖項目によって表される1つまたは複数の先祖ECを複製して、関連する先祖項目にアクセスせずに、前記文字列中の先祖ECを出力できるようにするステップとをさらに含む(42)記載の圧縮済みレコード展開方法。

【0309】(50)索引記号で位置指定された項目が文字列中のすべての先祖ECを含むことができないときに、文字列中の先祖項目の1つまたは複数の先祖ECを先祖項目中で複製して、他の先祖項目にアクセスせずに、さらに先祖ECを出力できるようにするステップをさらに含む(49)記載の圧縮済みレコード展開方法。

【0310】(51)前記辞書文字列中に複数の先祖文字が存在するとき、展開辞書内の一連の辞書項目のうち、第1先祖項目を除く各先祖辞書項目中に先祖ポイント・フィールドを提供するステップをさらに含む(50)記載の圧縮済みレコード展開方法。

【0311】(52)前記展開辞書項目中の記号長フィールド(SL)を、その項目に割り当てられたECからその文字列を開始するECまでの、辞書文字列中のECの連続したシーケンスにオスフニ...プレ 相レコードを今示

ッファ内に存在しているかどうかをSLから判定するステップとをさらに含む(49)記載の圧縮済みレコード展開方法。

【0312】(53) 対応する未圧縮レコード中で必要なECシーケンスと逆の順序である連鎖順序で、アクセスされる連鎖展開辞書項目から、索引記号で位置指定されたレコード文字列用のECを出力するステップと、未圧縮レコード中で必要な順序にECが並ぶ順序で、アクセスされる展開辞書項目の連鎖から出力されるECを、出力バッファ内に記録するステップと、対応する未圧縮レコード中で必要なシーケンスで、各項目内のECを選択するステップとをさらに含む(52)記載の圧縮済みレコード展開方法。

【0313】(54) 展開プロセスのために文字列中のすべてのECを取り出すために先祖項目への追加アクセスが必要でないことを示すために、関連するレコード文字列のすべてのEC(項目に割り当てられたECと先祖EC)が項目中に含まれるかどうかを項目中で示すステップとをさらに含む(52)記載の圧縮済みレコード展開方法。

【0314】(55) 展開辞書項目中のオフセット・フィールドを、先祖項目からさらにECを取り出すとき、出力バッファ内の現カーソル位置から順に未圧縮レコードが作成されるように、項目に割り当てられたECと項目中の先祖ECが配置される、出力バッファ内の現カーソル位置に対する位置を示す構造にするステップとをさらに含む(49)記載の圧縮済みレコード展開方法。

【0315】(56) 対応する未圧縮レコード中で必要なECシーケンスと逆の順序である連鎖順序で、アクセスされる連鎖展開辞書項目から、索引記号で位置指定されたレコード文字列のECを出力するステップと、未圧縮レコード中で必要な順序にECが並ぶ順序で、アクセスされる展開辞書項目の連鎖から出力されるECを、出力バッファ内に記録するステップと、対応する未圧縮レコード中で必要なシーケンスで、各項目内のECを選択するステップとをさらに含む(55)記載の圧縮済みレコード展開方法。

【0316】(57) 展開プロセスのために文字列中のすべてのECを取り出すために先祖項目への追加アクセスが必要でないことを示すために、関連するレコード文字列のすべてのEC(項目に割り当てられたECと先祖EC)が項目中に含まれるかどうかを項目中で示すステップとをさらに含む(55)記載の圧縮済みレコード展開方法。

【0317】(58) 関連するレコード文字列のすべてのECが項目中に含まれているわけでないことが項目中ですでに示されているときに、項目から取り出して出力バッファ内に置くべきECの数(項目に割り当てられた

【0318】(59) 記号が真に辞書項目を指定する索引記号であるかどうか、あるいは、記号が、ZL圧縮の使用時に、辞書内で、良好な圧縮を提供するのに十分な長さのZL文字列によって表されなかったために、未圧縮形式で現れる後続の該未圧縮文字の数のカウントを含むかどうかを示す標識で、索引記号を展開するステップと、上記カウントによって示される未圧縮文字の数を圧縮済みレコードから未圧縮レコードに移すことによって、展開プロセス中に、この標識で索引記号を処理するステップとをさらに含む(1)記載のレコード圧縮方法。

【0319】(60) ZL辞書文字列中の辞書項目を、1つの割り当てられたECおよび複数(N個)の子を持つ第1の種類のレベルの項目と、1つまたは複数の割り当てられたECを持つが、必ずしも2つ以上の子を持たない第2の種類のレベルの項目との、2種類の交互のレベルを持つ構造にするステップと、ここで定義する構造にしなかった場合よりもアクセスする必要のある辞書項目の数を減らすことによって、同一の文字が繰り返される、多数の異なる長さの文字列を圧縮する目的で、第2の種類のレベルの項目を、さらに、そのレベルにある第1の子がN個の割り当てられたECを持ち、次の子がN-1個の割り当てられたECを持ち、その次の子がN-2個の割り当てられたECを持ち、以下同様にして最後の子が1つの割り当てられたECを持つ構造にするステップとをさらに含む(3)記載のレコード圧縮方法。

【0320】(61) Ziv-Lempel(ZL)アルゴリズムを使用して、比較的大規模なデータベース内で未圧縮レコードを圧縮する方法と、圧縮済みレコードを未圧縮レコードに展開する方法を組み合わせた方法であって、データベースに前記ZLアルゴリズムを使用するコンピュータ・プログラムを適用して、前記データベース内のすべてのZL文字列を辞書文字列として含む辞書を前記データベースから生成することにより、前記データベースのレコードに対して圧縮操作と展開操作の両方を実行するための辞書項目を含む静的辞書を事前に生成するステップと、別のコンピュータ・プログラムを実行して、未圧縮レコード中の一連の文字を前記辞書中の前記辞書文字列と突き合わせることで、前記未圧縮レコード中のレコード文字列を検出するステップと、前記辞書中の辞書文字列と突き合わされるレコード文字列の、前記辞書内での終了位置を表す索引記号を出力して、前記未圧縮レコードに対応する圧縮済みレコードを提供するステップとにより、前記データベース内の未圧縮レコードが変更されているか否かにかかわらず、前記辞書を変更せずに、アクセスされた未圧縮レコードから圧縮済みレコードを生成するステップと圧縮済みレコード中にありかつ未圧縮レコード中のレコード文字列を表

を実行するステップと、見つかった辞書項目から各索引記号で表される各辞書文字列用の文字に前記アクセスするステップと、圧縮済みレコードから入力される各索引記号用のアクセスされた文字を出力して、対応する未圧縮レコードを再構築するステップとによって、前記辞書が事前生成された後に変更された未圧縮レコードから前記データベース内の圧縮済みレコードが生成されたかどうかにかかわらず、前記辞書を変更せずに、入力された圧縮済みレコードから未圧縮レコードを再生成するステップとを含む前記方法。

【0321】(62) コンピュータ・システムの記憶域に辞書を記憶するステップと、各辞書項目ごとに、コンピュータ・システムの記憶域からアクセス可能な記憶域アクセス・ユニットのサイズに等しい固定サイズを選択するステップとをさらに含む(61)記載のプロセスを使用する方法。

【0322】(63) 前記データベース内の各文字列を、各項目に前記辞書文字列中の1つまたは複数の当該の拡張文字(EC)が割り当てられ、かつ前記割り当てられたECが前記辞書文字列の真拡張文字(TEC)と呼ばれる、1つまたは複数の辞書項目によって表される辞書文字列として構造化するステップと、前記辞書中の各辞書文字列中の前記辞書項目を順方向および逆方向に連鎖するステップとをさらに含む(61)記載のプロセスを使用する方法。

【0323】(64) 子ECに割り当てられた辞書項目にアクセスせずに子文字(CC)からレコード文字列を検出できるとき、圧縮済みレコードを生成するためのプロセスにおける辞書の記憶域アクセスを削減するため、かつ辞書文字列から未圧縮レコードを再生成するプロセスにおける辞書の記憶域アクセスを削減するために、任意の辞書項目中で、同一の辞書文字列中の辞書項目に割り当てられたECの後または前に、子文字(CC)と呼ばれる、1つまたは複数のECの複製を作成するステップをさらに含む、(63)記載のプロセスを使用する方法。

【0324】(65) レコード文字列中の第1の文字を表す辞書項目に入り、未圧縮レコードから順次得られる後続の各レコード文字またはレコード文字の文字列を同一の辞書文字列中の後続の辞書項目と比較してレコード文字列の終りを決定し、辞書文字列中の最後の項目を検出するか、あるいは辞書文字列中のECと一致するレコード文字の後に続く次のレコード文字と一致しない辞書文字列中のECを検出することによって、レコード文字列の終りを見つけることにより、未圧縮レコードにおける各レコード文字列を検出するコンピュータ・プログラムを実行するステップをさらに含む(63)記載のプロセスを使用して圧縮済みレコードを生成する方法。

【0325】(66) 静的辞書を使用して、データベース内の未圧縮レコードから、静的辞書を使って、レコード圧縮速度を高めるために辞書をデータベースの変更に合わせて、圧縮済みレコードを生成するステップをさらに含む(61)記載のプロセスを使用する方法。

ドが変更され、あるいはデータベースに追加された、データベース内の未圧縮レコードから、静的辞書を使って、レコード圧縮速度を高めるために辞書をデータベースの変更に合わせて、圧縮済みレコードを生成するステップをさらに含む(61)記載のプロセスを使用する方法。

【0326】(67) 未圧縮レコードが変更された場合でも、静的辞書を使って、辞書をデータベースの変更に合わせて、未圧縮レコードを圧縮するステップをさらに含む(61)記載のプロセスを使用する方法。

【0327】(68) 適応辞書を使用する場合には実行できない方式でレコードを圧縮するために、データベースを任意の順序で(レコードを取り出した順序と同じランダムな順序、またはそれと異なるランダムな順序で、あるいは順次に)走査することによって、静的辞書が事前に生成された場合でも、該辞書を使って、データベースからランダムな順序で取り出された未圧縮レコードを圧縮するステップをさらに含む(61)記載のレコードを圧縮する方法。

【0328】(69) 圧縮済みレコード中の索引記号を使って辞書にアクセスして、未圧縮レコードを表す索引記号で表される文字列を取り出すことにより、辞書を使って圧縮された圧縮済みレコードを展開するステップとをさらに含む(61)記載のプロセスを使用する方法。

【0329】(70) 静的辞書を使って圧縮済みレコードを生成した後、記憶媒体内のデータベースに各圧縮済みレコードを記憶するステップと、記憶媒体内のデータベースから圧縮済みレコードにアクセスするステップと、静的辞書を使って、適応型の辞書を使用する場合よりも速い速度で、レコードを展開するステップとをさらに含む(69)記載のプロセスを使用する方法。

【0330】(71) それぞれ前記辞書文字列中に1つまたは複数の先祖ECを有する子ECを前記辞書文字列中に有する次の辞書項目を位置指定するために、辞書文字列の少なくとも第1の項目に子ポインタを記憶するステップをさらに含む(63)記載のプロセスを使用する方法。

【0331】(72) 第1の辞書項目で表されるアルファベットECの値によって示される位置に各辞書文字列の各第1辞書項目を位置指定し、各第1辞書項目をアルファベット項目として指定し、他のすべての辞書項目を非アルファベット項目として指定するが、各アルファベット項目の位置が、割り当てられたアルファベットECと関連付けられているためにどのアルファベット項目中にもECを必要としないステップをさらに含む(71)記載のプロセスを使用する方法。

【0332】(73) 前記静的辞書を、ECおよび制御フィールドを含む固定長項目を含む構造にする構造化システムをさらに含む(72)記載のプロセスを使用する方法。

【0333】(74) 同一の辞書文字列中の直接の先祖項目中の子ポインタによって非アルファベット辞書項目を位置指定するステップをさらに含む(73) 記載のプロセスを使用する方法。

【0334】(75) 前記構造化ステップが、非アルファベット項目中の制御フィールドを、項目に割り当てられた真ECの数を示すECカウント指示を含む構造にするステップをさらに含む(73) 記載のプロセスを使用する方法。

【0335】(76) 前記構造化ステップが、非アルファベット項目中で、制御フィールド中のカウント指示を、同一の辞書文字列用の先祖ESおよび割り当てられたECを該項目がいつ含むかを示す構造にするステップと、制御フィールドによっても割り当てられたECによっても使用されないスペースに項目の先祖ECを記録するステップとをさらに含む(73) 記載のプロセスを使用する方法。

【0336】(77) 前記構造化ステップが、アルファベット項目または非アルファベット項目の前記制御フィールドを、前記親項目に割り当てられたECの後に続く、辞書文字列中の文字である子EC(CCと呼ぶ)をいつ前記親項目が含むかを示す子標識を含む親辞書項目(親項目)として構造化し、それによって、CCに割り当てられた前記子辞書項目への追加のアクセスなしで、レコード文字列の終りを検出できるようにすることにより、CCが前記辞書内のレコード文字列の検出効率を向上できるようにするステップをさらに含む(73) 記載のプロセスを使用する方法。

【0337】(78) 親項目中に含まれるCCの数を示すカウント標識を親項目中に置き、各CCが、関連する子項目に割り当てられた第1のECであるか、または関連する子項目に割り当てられた1組の第1ECであることを事前決定し、あるいは項目中で指示するステップをさらに含む(77) 記載のプロセスを使用する方法。

【0338】(79) 1つまたは複数の子辞書項目(子項目)を有する子リストを含む、親項目の子辞書項目を提供するステップと、前記子リスト中の前記第1の子項目に対して所定の位置に、前記子リスト中の各子項目を位置指定するステップと、前記親項目中の各CCを前記子リスト中の当該の子項目に割り当てるステップとをさらに含む(77) 記載のプロセスを使用する方法。

【0339】(80) 前記辞書内の前記子リストを前記親項目中の子ポインタ・フィールドによって位置指定するステップをさらに含む(79) 記載のプロセスを使用する方法。

【0340】(81) 現在検出中のレコード文字列用の索引記号を生成するために、子ポインタと、前記辞書文字列中のECと突き合わされる次のレコード文字と一致する前記親項目中の関連CCの前記位置から、子辞書

載のプロセスを使用する方法。

【0341】(82) 親項目の制御フィールド中のmore-children標識をオンにセットして、前記親項目内に含まれるCCの数よりも多くの子項目が前記親項目の子リスト中にあることを示すステップをさらに含む(79) 記載のプロセスを使用する方法。

【0342】(83) 子検査標識を親項目中のCCと関連付けて、CCが現レコード文字と一致するときに、他の辞書項目にアクセスする必要なく、CCが辞書文字列を終了する(したがって、レコード文字列の終りを検出する)かどうかを示すステップをさらに含む(77) 記載のレコード圧縮方法。

【0343】(84) 子が指定CC以外に割り当てられたECを持たず、かつそれ自体の子を持たないことが事前決定されているとき、それ以上突合せプロセスを続行できないので、関連する子項目(子)へのアクセスが必要でないことを示すように、CCと関連する子検査標識を設定するステップ、または子が指定CC以外に割り当てられたECを持ち、あるいはそれ自体の子を持つことが事前決定されているとき、子に真の一致が見られるか否か、あるいは突合せプロセスが続行できないかどうか分からないので、子へのアクセスが必要であることを示すように、子検査標識を設定するステップをさらに含む(83) 記載のレコード圧縮方法。

【0344】(85) more-children標識を含む親項目(親)の下にある子リスト中の子項目(子)中に、親中の利用可能なスペースに、親中のCCと区別するために兄弟文字(SC)と呼ばれるその当該の各識別CCを含めることができない、同一の親の子供と関連するCCを置き、各SCが同一の親の関連する子に割り当てられた第1のECであるか、または同一の親の関連する子に割り当てられた1組の第1ECであることが事前決定され、あるいは項目中で指示されており、それによって、SCに割り当てられた子辞書項目への追加のアクセスなしで、レコード文字列の終りを検出できるようにすることにより、SCが辞書内のレコード文字列の検出効率を向上できるようにするステップと、子項目に含まれるSCの数を示すカウント標識を子項目中に置くステップと、SCを含む子項目中の各SCを、子リスト中の当該の子項目と関連付けられるように割り当てるステップとをさらに含む(82) 記載のレコード圧縮方法。

【0345】(86) SCを含む子項目を、親中に含まれる最後のCCによって指定される子項目の位置に対して所定の方式で位置指定するステップをさらに含む(85) 記載のレコード圧縮方法。

【0346】(87) 現在検出中のレコード文字列用の索引記号を生成するために、SCを含む第1の子項目の位置と、辞書文字列中のECと突き合わされる次のレコード文字と一致する第1の子項目中の関連するSCの位

らに含む(86)記載のレコード圧縮方法。

【0347】(88) SCを含む子項目の制御フィールド中のmore-siblings標識をオンにセットして、子項目が、子項目内に含まれるよりも多くのSCを有することを示すステップと、SCを含む第1の子項目中の最後のSCによって指定される子項目の位置に対して所定の位置に、追加SCを含む別の子項目を置くステップとをさらに含む(86)記載のレコード圧縮方法。

【0348】(89) 子検査標識を子項目中のSCと関連付けて、SCが現レコード文字と一致するとき、他の辞書項目にアクセスする必要なしに、SCが辞書文字列を終了する(したがって、レコード文字列の終りを検出する)かどうかを示すステップをさらに含む(85)記載のレコード圧縮方法。

【0349】(90) 子が指定SC以外に割り当てられたECを持たず、かつそれ自体の子を持たないことが事前決定されているとき、それ以上突合せプロセスが実行できないので、関連する子項目(子)へのアクセスが必要でないことを示すように、SCと関連する子検査標識を設定するステップ、または子が指定SC以外の割り当てられたECを持ち、あるいはそれ自体の子を持つことが事前決定されているとき、子に真の一致が見られるかどうか、あるいは突合せプロセスが実行できないかどうか分からないので、子へのアクセスが必要であることを示すように、子検査標識を設定するステップをさらに含む(89)記載のレコード圧縮方法。

【0350】(91) more-children標識を含む親項目(親)の下にある子リスト中に、アルファベット項目でも非アルファベット項目でもなく、前記親中の利用可能な前記スペースに、前記親中のCCと区別するために兄弟文字(SC)と呼ばれるその当該の各識別CCを含めることができない、前記同一の親の子項目(子)と関連する前記CCを含む、兄弟記述子(SD)と呼ばれる項目を置き、各SCが前記同一の親の関連する子項目

(子)に割り当てられた前記第1のECであるか、または前記同一の親の関連する子に割り当てられた1組の第1ECであることが事前決定され、あるいはSD中で指示されており、それによって、SCに割り当てられた前記子辞書項目への追加アクセスなしで、レコード文字列の終りを検出できるようにすることにより、SCが前記辞書内のレコード文字列の検出効率を向上できるようにするステップと、前記SDに含まれるSCの数を示すカウント標識を前記SD中に置くステップと、前記SD中の各SCを、前記子リスト中の当該の子項目と関連付けられるように割り当てるステップとをさらに含む(82)記載のレコード圧縮方法。

【0351】(92) SDを、親中に含まれる最後のCCによって指定される子項目の位置に対して所定の方式

【0352】(93) 現在検出中のレコード文字列用の索引記号を生成するために、SCを含むSDの位置と、辞書文字列中のECと突き合わされる次のレコード文字と一致するSD中の関連するSCの位置とから、子項目の位置を算出するステップをさらに含む(92)記載のレコード圧縮方法。

【0353】(94) SDの制御フィールド中のmore-siblings標識をオンにセットして、SDが、SD内に含まれるよりも多くのSCを有することを示すステップと、第1のSD中の最後のSCによって指定される子項目の位置に対して所定の位置に、上記の追加SCを含む別のSDを置くステップとをさらに含む(92)記載のレコード圧縮方法。

【0354】(95) 子検査標識をSD中のSCと関連付けて、SCが現レコード文字と一致するとき、他の辞書項目にアクセスする必要なしに、SCが辞書文字列を終了する(したがって、レコード文字列の終りを検出する)かどうかを示すステップをさらに含む(91)記載のレコード圧縮方法。

【0355】(96) 子が指定SC以外に割り当てられたECを持たず、かつそれ自体の子を持たないことが事前決定されているとき、それ以上突合せプロセスが実行できないので、関連する子項目(子)へのアクセスが必要でないことを示すように、SCと関連する子検査標識を設定するステップ、または子が指定SC以外の割り当てられたECを持ち、あるいはそれ自体の子を持つことが事前決定されているとき、子に真の一致が見られるかどうか、あるいは突合せプロセスが実行できないかどうか分からないので、子へのアクセスが必要であることを示すように、子検査標識を設定するステップをさらに含む(95)記載のレコード圧縮方法。

【0356】(97) 辞書項目における終了の指示が、辞書文字列の終りであることを示すとき、辞書項目内のすべてのECがレコード文字と一致するとき、レコード文字列を検出するステップをさらに含む(61)記載のレコード圧縮方法。

【0357】(98) 展開中の圧縮済みレコードが、静的展開辞書を事前生成した後に変更された対応する未圧縮レコードを表す場合でも、静的辞書を使用してデータベースの圧縮済みレコードを展開することによって、静的辞書に対応する未圧縮レコードに適合させなくて済むようにするステップをさらに含む(61)記載のプロセスを使用する方法。

【0358】(99) 適応辞書を使用しては実行できない方式でレコードを圧縮し展開するために、データベースを任意の順序で(データベースから取り出した順序と同じランダムな順序、またはそれと異なるランダムな順序で、あるいは順次に)走査することにより、静的辞書

コードを展開するステップをさらに含む(61)記載の圧縮済みレコード展開方法。

【0359】(100) 適応辞書を使用しては実行できない方式でレコードを圧縮し展開するために、データベースから任意の順序で(データベースから取り出した順序と同じランダムな順序、またはそれと異なるランダムな順序で、あるいは順次に)取り出した未圧縮レコードを圧縮することによって圧縮済みレコードが生成された場合でも、静的辞書を使用して、データベースからランダムな順序で取り出した圧縮済みレコードを展開するステップをさらに含む(61)記載の圧縮済みレコード展開方法。

【0360】(101) データベースのレコードを圧縮および展開するために、ネットワークの受信端と送信端で辞書を記憶するステップをさらに含む(61)記載のプロセスを使用する方法。

【0361】(102) ネットワークの受信端で静的展開辞書を使用して圧縮済みレコードを展開し、各未圧縮レコードの伝送と供に辞書を伝送することなしに、対応する未圧縮レコードを生成するステップをさらに含む(61)記載のプロセスを使用する方法。

【0362】(103) 索引記号で位置指定された項目内に含まれるよりも多くの先祖EC(位置指定された項目の先祖に割り当てられたEC)が、関連する辞書文字列中に存在するとき、索引記号で位置指定された項目に連鎖された、辞書内の1つまたは複数の先祖辞書項目を提供するステップと、索引記号で位置指定された項目中に、先祖項目によって表される1つまたは複数の先祖ECを複製して、関連する先祖項目にアクセスせずに、文字列中の先祖ECを出力できるようにするステップをさらに含む(61)記載のプロセスを使用する方法。

【0363】(104) 索引記号で位置指定された項目が文字列中のすべての先祖ECを含むことができないときに、文字列中の先祖項目の1つまたは複数の先祖ECを先祖項目中で複製して、他の先祖項目にアクセスせずに、さらに先祖ECを出力できるようにするステップをさらに含む(103)記載のプロセスを使用する方法。

【0364】(105) 辞書文字列中に複数の先祖文字が存在するとき、展開辞書内の一連の辞書項目のうち、第1先祖項目を除く各先祖辞書項目中に先祖ポインタ・フィールドを提供するステップをさらに含む(104)記載の圧縮済みレコード展開方法。

【0365】(106) 辞書項目中の記号長フィールド(SL)を、その項目に割り当てられたECからその文字列を開始するECまでの、辞書文字列中のECの総数を示す構造にするステップと、現レコード文字列用に出力される文字を受け取るためのスペースが出力バッファ内に存在しているかどうかをSLから判定するステップをさらに含む(61)記載のプロセスを使用する方法。

【0366】(107) 対応する未圧縮レコード中で必要なECシーケンスと逆の順序である連鎖順序で、アクセスされる連鎖辞書項目から、索引記号で位置指定されたレコード文字列用のECを出力するステップと、未圧縮レコード中で必要な順序にECが並ぶ順序で、アクセスされる辞書項目の連鎖から出力されるECを、出力バッファ内に記録するステップと、対応する未圧縮レコード中で必要なシーケンスで、各項目内のECを選択するステップとをさらに含む(106)記載のプロセスを使用する方法。

【0367】(108) 展開プロセスのために文字列中のすべてのECを取り出すために先祖項目への追加アクセスが必要でないことを示すために、関連するレコード文字列のすべてのEC(項目に割り当てられたECと先祖EC)が項目中に含まれるかどうかを項目中で示すステップをさらに含む(106)記載のプロセスを使用する方法。

【0368】(109) 辞書項目中のオフセット・フィールドを、先祖項目からさらにECを取り出すとき、出力バッファ内の現カーソル位置から順に未圧縮レコードが作成されるように、項目に割り当てられたECと項目中の先祖ECが配置される、出力バッファ内の現カーソル位置に対する位置を示す構造にするステップをさらに含む(61)記載の、圧縮済みレコード展開方法。

【0369】(110) 対応する未圧縮レコード中で必要なECシーケンスと逆の順序である連鎖順序で、アクセスされる連鎖辞書項目から、索引記号で位置指定されたレコード文字列のECを出力するステップと、未圧縮レコード中で必要な順序にECが並ぶ順序で、アクセスされる辞書項目の連鎖から出力されるECを、出力バッファ内に記録するステップと、対応する未圧縮レコード中で必要なシーケンスで、各項目内のECを選択するステップとをさらに含む(109)記載の、圧縮済みレコード展開方法。

【0370】(111) 展開プロセスのために文字列中のすべてのECを取り出すために先祖項目への追加アクセスが必要でないことを示すために、関連するレコード文字列のすべてのEC(項目に割り当てられたECと先祖EC)が項目中に含まれるかどうかを項目中で示すステップをさらに含む(109)記載の圧縮済みレコード展開方法。

【0371】(112) 関連するレコード文字列のすべてのECが項目中に含まれているわけでないことが項目中ですでに示されているときに、項目から取り出して出力バッファ内に置くべきECの数(項目に割り当てられたECと先祖EC)を項目中で示すステップをさらに含む(111)記載の圧縮済みレコード展開方法。

【0372】(113) 記号が真に辞書項目を指定する索引記号であることを示すために、索引記号が71に始

な長さのZL文字列によって表されなかったために、未圧縮形式で現れる後続の該未圧縮文字の数のカウントを含むかどうかを示す標識で、索引記号を展開するステップと、上記カウントによって示される未圧縮文字の数を圧縮済みレコードから未圧縮レコードに移すことによって、展開プロセス中に、この標識で索引記号を処理するステップとをさらに含む(61)記載のレコード圧縮方法。

【0373】(114) ZL辞書文字列中の辞書項目を、1つの割り当てられたECおよび複数(N個)の子を持つ第1の種類のレベルの項目と、1つまたは複数の割り当てられたECを持つが、必ずしも2つ以上の子を持たない第2の種類のレベルの項目との、2種類の交互のレベルを持つ構造にするステップと、ここで定義する構造にしなかった場合よりもアクセスする必要がある辞書項目の数を減らすことによって、同一の文字が繰り返される、多数の異なる長さの文字列を圧縮する目的で、第2の種類のレベルの項目を、さらに、そのレベルにある第1の子がN個の割り当てられたECを持ち、次の子がN-1個の割り当てられたECを持ち、その次の子がN-2個の割り当てられたECを持ち、以下同様にして最後の子が1つの割り当てられたECを持つ構造にするステップとをさらに含む(61)記載のレコード圧縮方法。

【0374】(115) 関連するレコード文字列のすべてのECが項目中に含まれているわけでないことが項目中ですでに示されているときに、項目から取り出して出力バッファ内に置くべきECの数(項目に割り当てられたECと先祖EC)を項目中で示すステップをさらに含む(54)記載の圧縮済みレコード展開方法。

【0375】(116) 関連するレコード文字列のすべてのECが項目中に含まれているわけでないことが項目中ですでに示されているときに、項目から取り出して出力バッファ内に置くべきECの数(項目に割り当てられたECと先祖EC)を項目中で示すステップをさらに含む(108)記載の圧縮済みレコード展開方法。

【図面の簡単な説明】

【図1】本願発明の好ましい実施例で使用する圧縮の概要を示す図である。

【図2】本願発明の好ましい実施例で使用する展開の概要を示す図である。

【図3】データの圧縮に使用する発信元バッファおよび宛先バッファを備えた従来のZiv-Lempel辞書を示す図である。

【図4】文字列のZiv-Lempelツリーの例を示す図である。

【図5】図4に示した文字ツリーを表すリスト形式の辞書を示す図である。

【図7】圧縮と展開の両方に使用される辞書を示す図である。

【図8】任意の辞書項目に先祖文字(PEC)を含む可能性のある辞書を示す図である。

【図9】任意の項目に子文字(CC)を含む可能性のある辞書を示す図である。

【図10】任意の項目に兄弟文字(SC)を含む可能性のある辞書を示す図である。

【図11】辞書のアルファベット項目(最初の256文字)で利用できる制御数字を示す図である。

【図12】非アルファベット項目(最初の256文字の後)で利用できる制御ビットを示す図である。

【図13】アルファベット項目における制御数字および子ポインタ(CPTR)を示す図である。

【図14】非アルファベット項目における制御バイトおよびCPTRを示す図である。

【図15】CPTRの代わりに先行ポインタ(PPTR)を示す図である。

【図16】すべて同一の非アルファベット項目に含まれる、制御バイト、CPTR、およびPPTRを示す図である。

【図17】辞書構造で利用できる複数の項目フォーマットの例、F1A、F1、F2、F3A、F3、F4を示す図である。

【図18】辞書項目F1、F2、F3A、F3、F4の使用例を示す図である。

【図19】図18の例に対応する文字記号ツリーを示す図である。

【図20】図18の項目フォーマットのツリーを示す図である。

【図21】図18の真拡張文字のツリーを示す図である。

【図22】図18のすべての拡張文字のツリーを示す図である。

【図23】子文字(CC)および兄弟文字(SC)を使用して辞書項目を位置指定する方法を示す図である。

【図24】図23の真拡張文字のツリーを示す図である。

【図25】兄弟記述子中の兄弟文字(SC)を使用して辞書項目を位置指定する方法を示す図である。

【図26】図25のツリー拡張文字のツリーを示す図である。

【図27】本願発明の好ましい実施例で使用され、圧縮または展開を実行するときに使用できる圧縮呼出し命令を示す図である。

【図28】圧縮呼出し命令が使用するレジスタを示す図である。

【図29】好ましい実施例の別個の圧縮用辞書で使

流れ図である。

【図31】好ましい実施例で使用される圧縮プロセスの流れ図である。

【図32】好ましい実施例の別個の展開用辞書で使われる項目のフォーマットを示す図である。

【図33】好ましい実施例で使用される展開プロセスの流れ図である。

【図34】圧縮辞書および展開辞書の例、特に、同一の

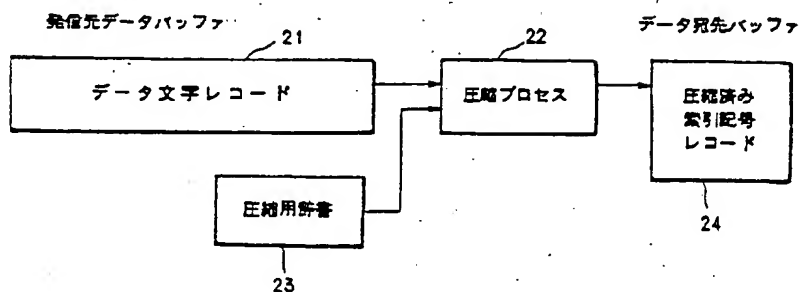
文字が繰り返される、多数の異なる長さの文字列を圧縮するための辞書を示す図である。

【図35】好ましい実施例に短記号オプションがどのように含まれるかを示す図である。

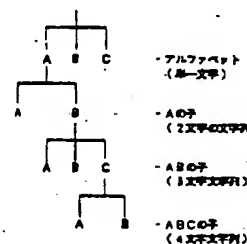
【図36】短記号オプションによって索引記号がどのようにして長記号に変更されるかを示す図である。

【図37】短記号と呼ばれる未圧縮文字を含む短記号文字列の例を示す図である。

【図1】

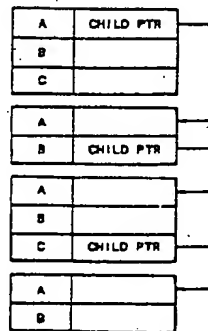
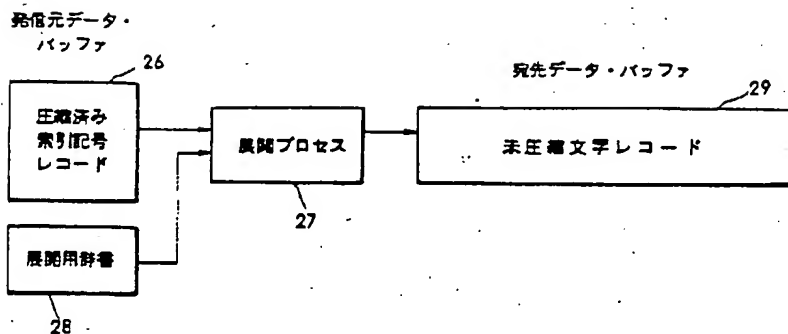


【図4】



【図5】

【図2】



【図6】

【図7】

【図8】

【図9】

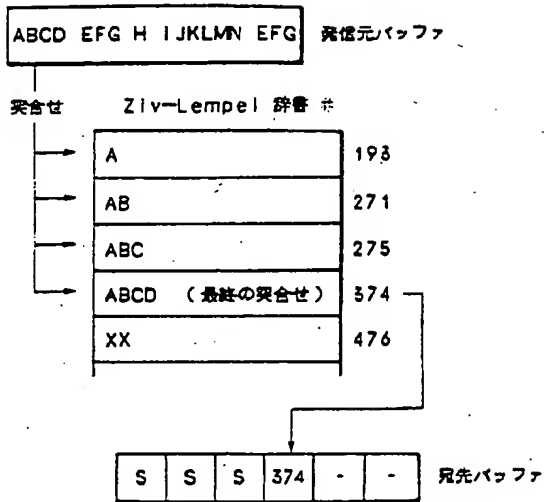
INDEX	EC	CPTR
0	A	5
1	B	
2	C	
3	A	
4	B	5
5	A	
6	B	
7	C	8
8	A	
9	B	

INDEX	EC	CPTR	PPTR
0	A	5	
1	B		
2	C		
3	A		8
4	B	5	0
5	A		4
6	B		4
7	C	8	4
8	A		7
9	B		7

INDEX	EC	CPTR	PPTR	文字列
0	A	5		A
1	B			B
2	C			C
3	AA			AA
4	AB	5		AB
5	ABA			ABA
6	ABD			ABD
7	ABC	8		ABC
8	A		7	ABCA
9	A		7	ABCB

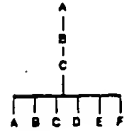
INDEX	EC	CPTR	PPTR	OC
0	A	5		AB
1	B			
2	C			
3	AA			
4	AB	5		ABC
5	ABA			
6	ABD			
7	ABC	8		AB
8	A		7	
9	B		7	

【図3】

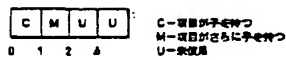


【図10】

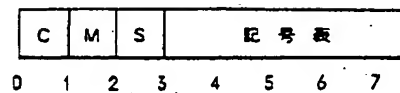
位置	EC フィールド	CPTR	PPTR	CC フィールド	SC フィールド
0	A	8		ABC	
1	B				
2	C				
3	AA				
4	AB				
5	AC				
6	AD				EFG
7	AE				
8	AF				
9	AG				
10	AH				I
11	AI				



【図11】

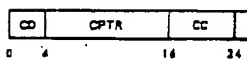


【図12】

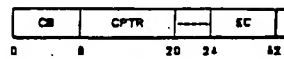


C-項目が子を持つ
M-項目がさらに子を持つ
S-項目がさらに兄弟を持つ

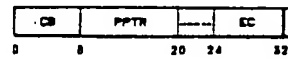
【図13】



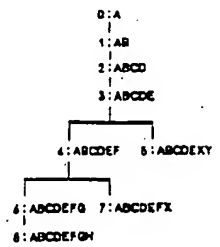
【図14】



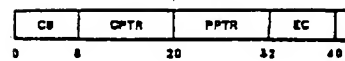
【図15】



【図19】



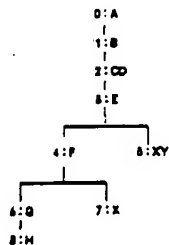
【図16】



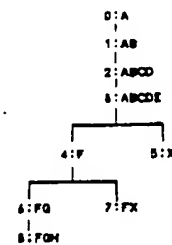
【図20】



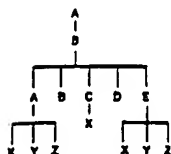
【図21】



【図22】



【図24】



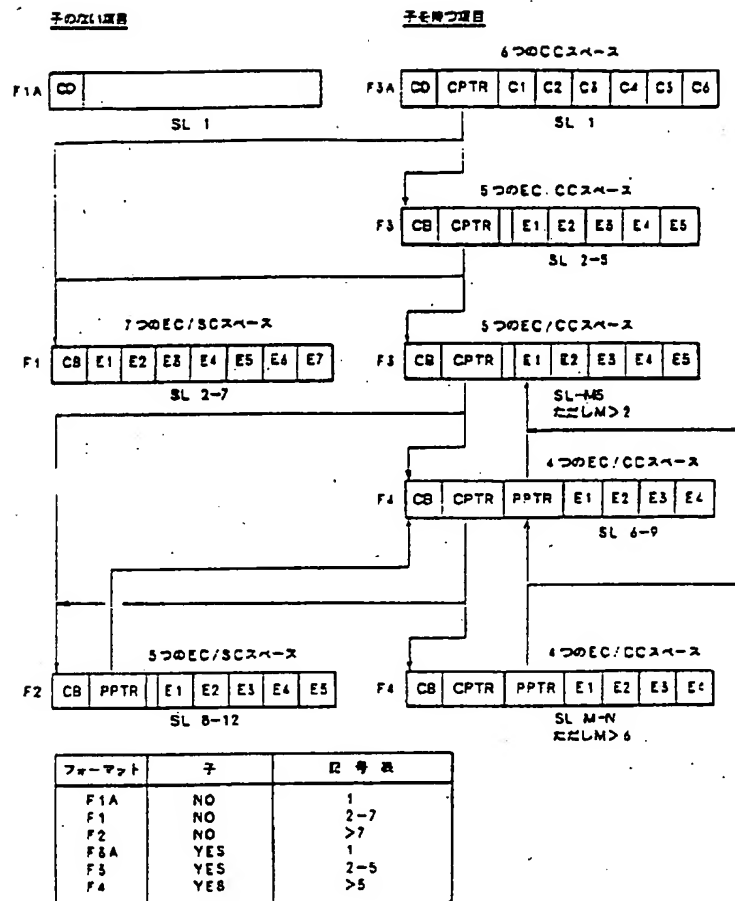
【図36】

← 最記号 →

最記号

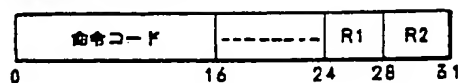
0 最記号

【図17】



【図27】

圧縮呼出し (CMPSC)



条件コード:

CC0 発信元オペランドの終りに到達

CC1 宛先オペランドの終りに到達、発信元オペランドに到達しない

【図18】

位置 FMT 呼称項目		文字記号							
0	F3A	CD	CPTR	C1	C2	C3	C4	C5	C6
		C,	1	B	B				
1	F3	CB	CPTR	---	E1	E2	C1	C2	C3
		C...2	2	--	A	B	C	C	
2	F3	CB	CPTR	---	E1	E2	E3	E4	C1
		C...4	3	---	A	B	C	D	E
3	F3	CB	CPTR	---	E1	E2	E3	E4	E5
		C,M...5	4	---	A	B	C	D	E
4	F4	CB	CPTR	PPTR	E1	C1	C2	C3	
		C...S,6	6	3	F	G	X	X	
5	F1	CB	E2	E2	E3	E4	E5	E6	E7
		...7	A	B	C	D	E	X	Y
6	F4	CB	CPTR	PPTR	E1	E2	C1	C2	
		C...7	8	3	F	G	H	H	
7	F1	CB	E2	E2	E3	E4	E5	E6	E7
		...7	A	B	C	D	E	F	X
8	F2	CB	PPTR	---	E1	E2	E3	E4	E5
		...8	3	---	F	G	H		
8	F1	CB	E2	E2	E3	E4	E5	E6	E7
		...7	A	B	C	D	E	X	Y

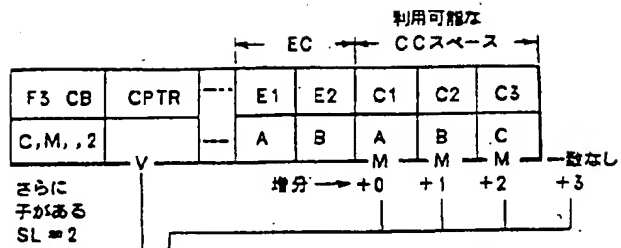
【図35】

GR1

呼称アドレス*		----	S	ISS	E	CBN
0	1	20	24	27	29	31

S=0: 短記号オプションなし
 S=1: 短記号オプション

【図23】

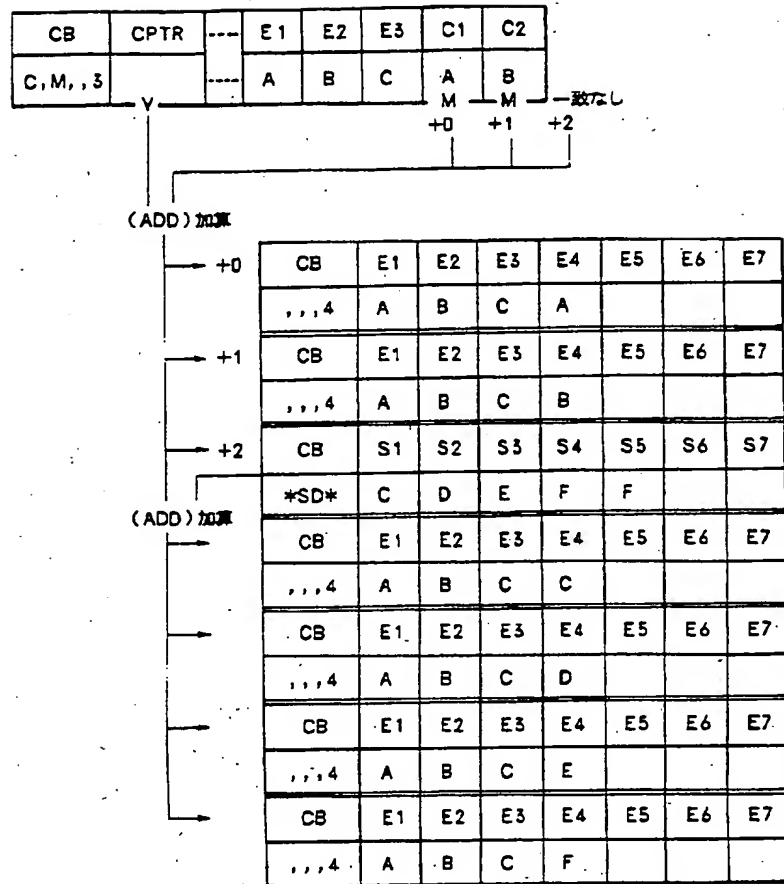


(ADD) 加算

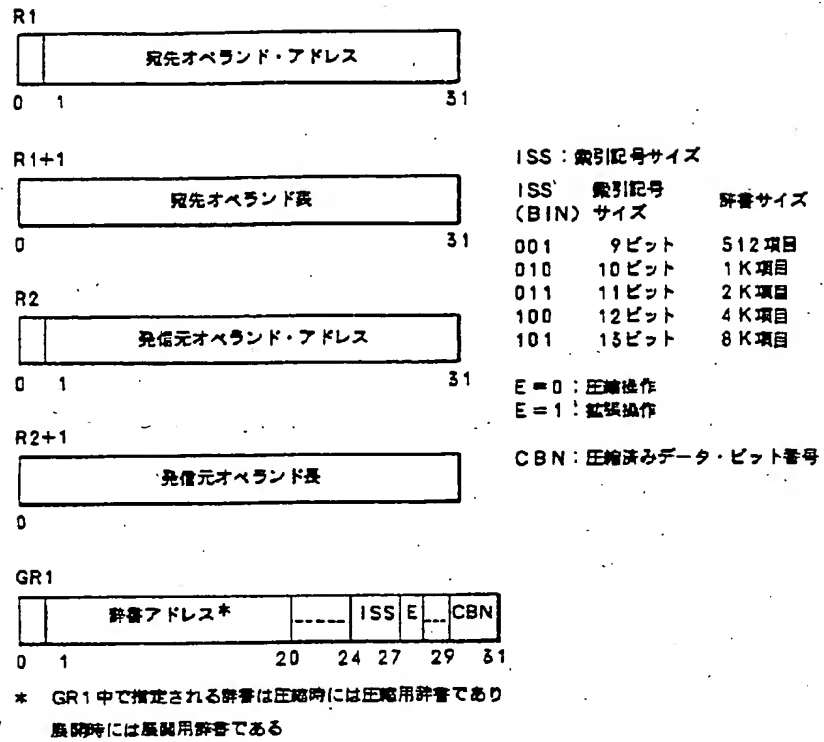
VNS = 値を示さない

→ +0	F3 CB	CPTR	---	E1	E2	E3	C1	C2
	C, M, , 3	VNS	---	A	B	A	X	Y
→ +1	F1 CB	E1	E2	E3	E4	E5	E6	E7
	, , , 3	A	B	B	B			
→ +2	F3 CB	CPTR	---	E1	E2	E3	C1	C2
	C, , , 3	VNS	---	A	B	C	X	X
→ +3	F1 CB	E1	E2	E3	S1	S2	S3	S4
	, , S, 3	A	B	D	E	E		
(ADD) 加算 → +4	F3 CB	CPTR	---	E1	E2	E3	C1	C2
	C, M, , 3	VNS	---	A	B	E	X	Y

【図25】



【図28】



【図29】

表23

ACT	AEC カウント: この項目中のAECの数
[AEC]	このフィールドに追加の拡張文字を入力することができる
...	最初のフィールドを繰り返すことができる
CC	子文字 子の第1 EC
CCT	CC カウント この項目中のCCの数
CPTR	子ポインタ 第1の子の索引
D	倍文字項目 ACT=0または1と同じ
M	more-childrenビット この項目はCCよりも多くの子を持つ
S	more-siblingsビット この項目はSCよりも多くの兄弟を持つ
SC	兄弟文字 兄弟の第1 EC
SCT	兄弟文字カウント この項目中のSCの数
X	CC用の子検査ビット 子はAECまたは子を持つ
Y	SC用の子検査ビット 子はAECまたは子を持つ

フォーマットC0文字項目 (CCT=0)

CCT	ACT	[AEC]				
0	3	8	11	24	32	40	48	56	63

CCT=0 CCを持たないフォーマットC0項目を示す

ACT=0ないし4 この項目には0ないし4つのAECを入れることができる

フォーマットC1文字項目 (CCT=1)

CCT	X	ACT	CPTR	[AEC]	...	CC			
0	3	8	11	24		N			63

CCT=1 1つのCCを持つフォーマットC1項目を示す

ACT=0ないし4 この項目には0ないし4つのAECを入れることができる

 $N = 24 + (ACT * 8)$

フォーマットCG1文字項目 (CCT>1)

CCT	XXXX	MD	CPTR	[AEC]	CC	CC	
0	3	8	11	24	N	N+8			63

CCT>1はCG1を示す D=0の場合はCCT=2ないし5、D=1の場合はCCT=2ないし4

D=0または1 この項目には0または1つのAECを入れることができる

 $N = 24 + (D * 8)$

XXXX はそれぞれこの項目中のCCSと関連付けられている

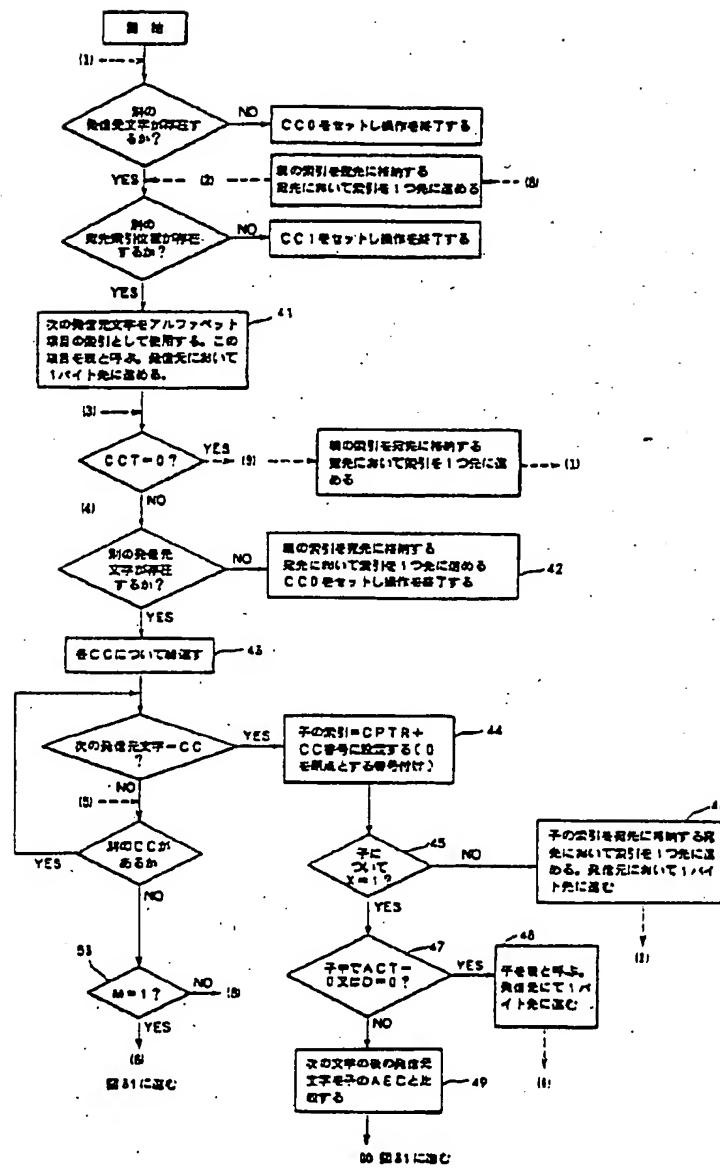
兄弟記述子項目

SCT YYYY YS	SC
0 8 8 10 16 24 32 40 48 56 63						

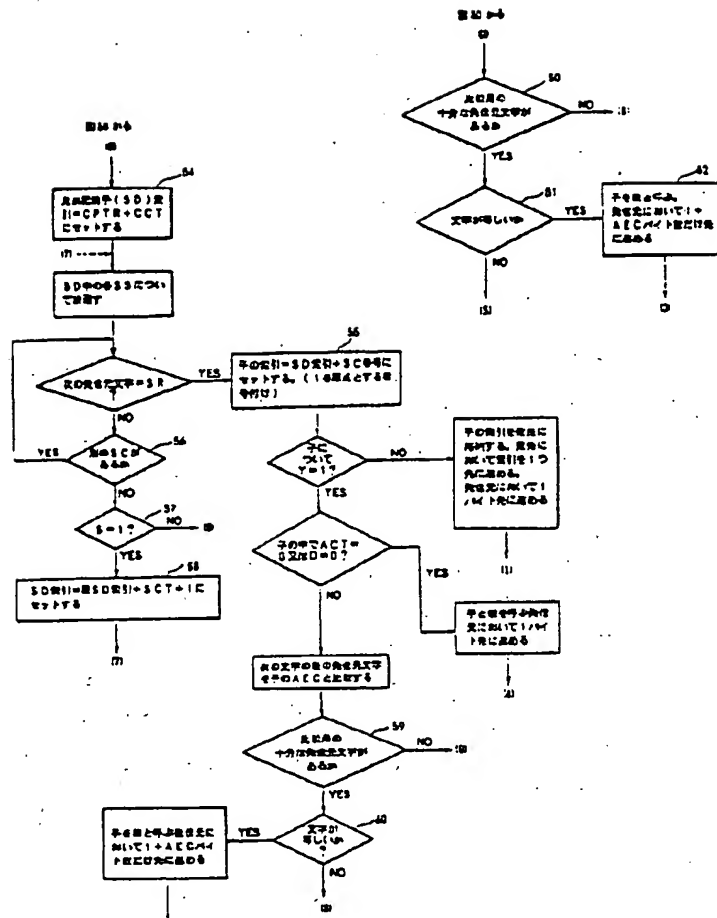
SCT=1ないし6

YYYYYY はそれぞれこの項目中のSCと関連付けられている

【図30】



【図3 1】



【図32】

表記法

- CSL 完全記号長 先行なし項目中のECの数
 EC 拡張文字
 --- 直前のフィールドを繰り返すことができる
 OFST この項目中の 1 ECを配置すべき場所に対するオフセット
 PPTR 先祖ポインタ 直前のECを含む項目の索引
 PSL 部分記号長 先行あり項目中のECの数

先行なし文字項目 (PSL = 0)

PSL	CSL	EC	---	---	---	---	---	---		
0	2	5	8	16	24	32	40	48	52	63

PSL = 0 は先行なし項目を示す

CSL = 1 ないし 7

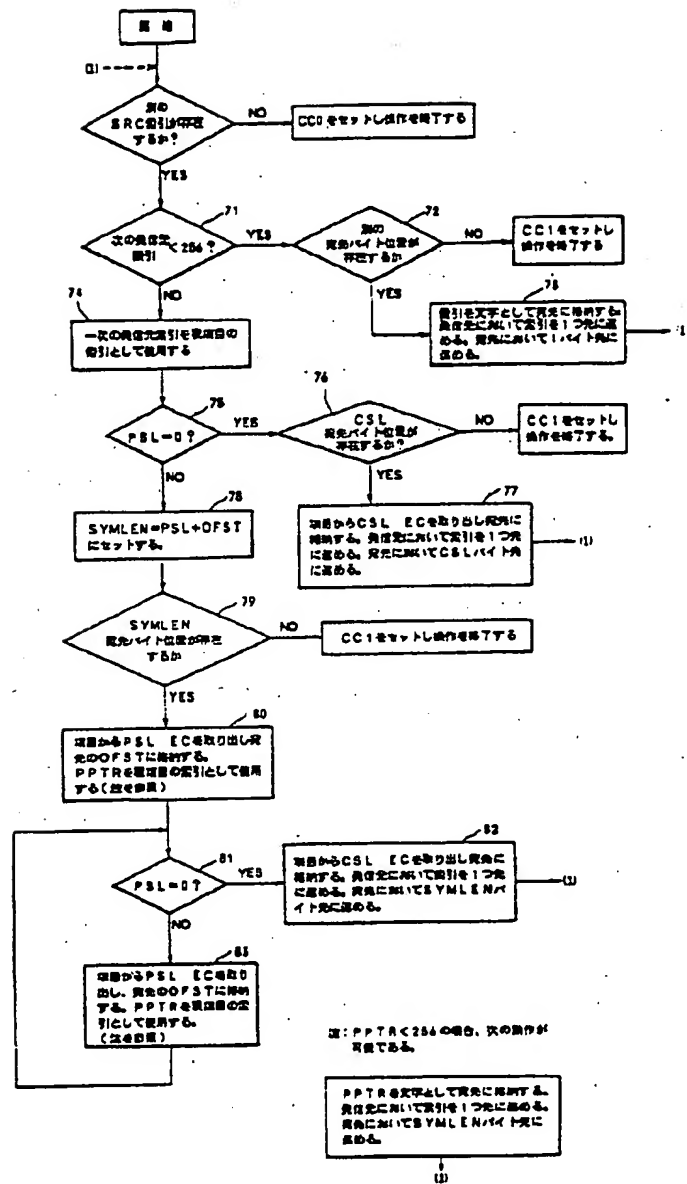
先行あり文字項目 (PSL > 0)

PSL	PPTR	EC	---	---	---	---	---	
0	3	16	24	32	40	48	52	63

PSL > 0 は先行あり項目を示す PSL = 1 ないし 5

OFST = 0 ないし 255

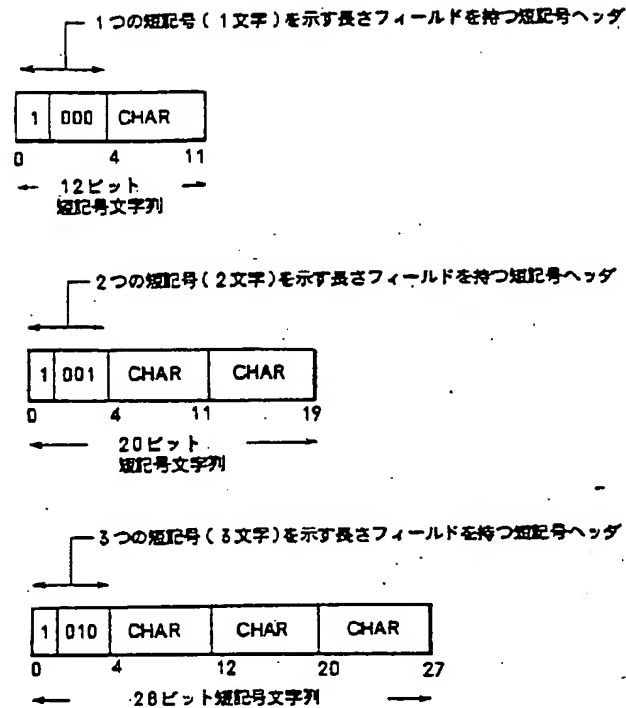
【図33】



【図34】

平綴用符号							星綴用符号						
索引	D OR					SIG							OFST
	CCT	XXXXX	ACT	CPTR	AECG	CCS	AS	REFS	PSL	CSL	PFTR	ECS	
...													
199	5	11110	0	256		AAAAA	1	1	0	1		A	
...													
256	1	1	4	261	AAAA	A	6	2	0	6		AAAAAA	
257	0		3		AAA		5	3	0	5		AAAAA	
258	0		2		AA		4	4	0	4		AAAA	
259	0		1		A		3	5	0	3		AAA	
260	0		0				2	6	0	2		AA	
261	5	11110	0	262		AAAAA	7	3	0	7		AAAAAAA	
262	1	1	4	267	AAAA	A	12	4	5		261	AAAAA	7
263	0		3		AAA		11	5	4		261	AAAA	7
264	0		2		AA		10	6	3		261	AAA	7
265	0		1		A		9	7	2		261	AA	7
266	0		0				8	8	1		261	A	7
267	5	11110	0	268		AAAAA	13	5	1		262	A	12
268	1	1	4	273	AAAA	A	18	6	1		269	A	17
269	0		3		AAA		17	7	5		262	AAAAA	12
270	0		2		AA		16	8	4		262	AAAA	12
271	0		1		A		15	9	3		262	AAA	12
272	0		0				14	10	2		262	AA	12
273	5	11110	0	274		AAAAA	19	7	2		269	AA	17
274	1	1	4	279	AAAA	A	24	8	2		276	AA	22
275	0		3		AAA		23	9	1		276	A	22
276	0		2		AA		22	10	5		269	AAAAA	17
277	0		1		A		21	11	4		269	AAAA	17
278	0		0				20	12	3		269	AAA	17
279	5	11110	0	280		AAAAA	25	9	3		276	AAA	22

【図37】



フロントページの続き

(72)発明者 クラーク・クルツ
アメリカ合衆国12528、ニューヨーク州ハ
イランド、ウッドランド・ドライブ 7

(72)発明者 ケンス・アーネスト・ブラムベック
アメリカ合衆国12603、ニューヨーク州ボ
ーキーブシー、デイジー・レーン 7

(72)発明者 バスカール・シンハ
アメリカ合衆国01921、メリーランド州ボ
ックスフォード、アンナズ・ウェイ 41